58th Annual IEEE Symposium on Foundations of Computer Science

Sample Efficient Estimation and Recovery in Sparse FFT via Isolation on Average

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

***Abstract*—The problem of computing the Fourier Transform of a signal whose spectrum is dominated by a small number k of frequencies quickly and using a small number of samples of the signal in time domain (the Sparse FFT problem) has received significant attention recently. It is known how to approximately compute the k-sparse Fourier transform in $\approx\ k\log^2 n$ time [Hassanieh et al'STOC'12], or using the optimal number $O(k\log n)$ of samples [Indyk et al'FOCS'14] in time domain, or come within $(\log\log n)^{O(1)}$ factors of both these bounds simultaneously, but no algorithm achieving the optimal $O(k\log n)$ bound in sublinear time is known.**

At a high level, sublinear time Sparse FFT algorithms operate by 'hashing' the spectrum of the input signal into $\approx k$ 'buckets', identifying frequencies that are 'isolated' in their buckets, subtracting them from the signal and repeating until the entire signal is recovered. The notion of 'isolation' in a 'bucket', inspired by applications of hashing in sparse recovery with arbitrary linear measurements, has been the main tool in the analysis of Fourier hashing schemes in the literature. However, Fourier hashing schemes, which are implemented via filtering, tend to be 'noisy' in the sense that a frequency that hashes into a bucket contributes a non-negligible amount to neighboring buckets. This leakage to neighboring buckets makes identification and estimation challenging, and the standard analysis based on isolation becomes difficult to use without losing $\omega(1)$ factors in sample complexity.

In this paper we propose a new technique for analysing noisy hashing schemes that arise in Sparse FFT, which we refer to as *isolation on average*. We apply this technique to two problems in Sparse FFT: estimating the values of a list of frequencies using few samples and computing Sparse FFT itself, achieving sample-optimal results in $k\log^{O(1)} n$ time for both. We feel that our approach will likely be of interest in designing Fourier sampling schemes for more general settings (e.g. model based Sparse FFT).

***Keywords*-Sparse Fourier Transform; Fourier sampling; sublinear algorithms**

I. Introduction

The Discrete Fourier Transform (DFT) is a fundamental computational primitive with numerous applications in areas such as digital signal processing, medical imaging and data analysis as a whole. The fastest known algorithm for computing the Discrete Fourier Transform of a signal of length n is the FFT algorithm, designed by Cooley and Tukey in 1965. The efficiency of FFT, which runs in time $O(n\log n)$ on any signal of length n, has contributed significantly to its popularity as a computational primitive, making FFT one of the top 10 most important algorithms of the 20th century [Cip00]. However, computational efficiency of FFT is not the only reason why the Fourier transform emerges in many applications: in signal processing the Fourier basis is often a convenient way of representing signals since it concentrates their energy on a few components, allowing compression (which is the rationale behind image and video compression schemes such as JPEG and MPEG), and in medical imaging applications such as MRI the Fourier transform captures the physics of the measurement process (the problem of reconstructing an image from MRI data is exactly the problem of reconstructing a signal x from Fourier measurements of x). While FFT works for worst case signals, signals arising in practice often exhibit structure that can be exploited to speed up the computation of the Fourier transform. For example, it is often the case that most of the energy of these signals is concentrated on a small number of components in Fourier domain. In other words, the signals that arise in applications are often *sparse* (have a small number of nonzeros) or *approximately sparse* (can be well approximated by a small number of dominant coefficients) in the Fourier domain. This motivates the question of (approximately) computing the Fourier transform of a signal that is (approximately) sparse in Fourier domain using *few samples of the signal in time domain (i.e. with small sample complexity)* and *small runtime*. We note that while runtime is a natural parameter to optimize, sample complexity is at least as important in applications such as medical imaging, where sample complexity governs the *measurement* complexity of the imaging process.

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 $\widehat{x}\in\mathbb{C}^n$, which is equivalent to the problem above (since the inverse Fourier transform only differs from the Fourier transform by a conjugation), but leads to somewhat more compact notation. This problem has been studied extensively. The seminal work of [CT06], [RV08] in *compressed sensing* first showed that length n signals with at most k 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), originating from computational complexity and learning theory, has resulted in algorithms that use $k\log^{O(1)} n$ samples and

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.66

$k \log^{O(1)} n$ runtime (i.e. the runtime is *sublinear* in 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], [LWC12], [BCG+12], [HAKI12], [PR13], [HKPV13], [IKP14], [IK14], [Kap16], [PS15], [CKPS16]. Nevertheless, despite significant progress that has recently been achieved, important gaps in our understanding of sample and time efficient recovery from Fourier measurements remain. We address some of these gaps in this work.

The main contribution of this work is a new technique for designing and analyzing sample efficient sublinear time Sparse FFT algorithms. We refer to this technique as *isolation on average*. We apply our technique to two problems in the area of Sparse Fourier Transform computation, namely estimation and recovery with Fourier measurements.

Our results on estimation: In the first problem we are given a subset $S \subseteq [n]$ of locations in the time domain and are asked to estimate x_S from a few values of $\widehat{x}$. Formally, we would like the algorithm to output a signal x' with $\mathrm{supp}(x') \subseteq S$ such that

$$\|x - x'\|_2^2 \leq (1+\epsilon)\|x_{[n]\setminus S}\|_2^2. \tag{1}$$

In other words, we would like to output an estimate x' of x that is correct up to the 'noise', i.e. elements outside of S (to achieve (1), it suffices to ensure that $\|(x-x')_S\|_2^2 \leq \epsilon\|x_{[n]\setminus S}\|_2^2$). Note that in some of the applications described above one often has a good prior on which coefficients of x are the dominant ones, and a natural question is whether one can recover the values of x_S quickly, using few samples, and in a noise robust manner, i.e. solve (1). Our main result on estimation is Algorithm 2 (presented in Section IV) together with

Theorem I.1. *For every $\epsilon \in (1/n, 1), \delta \in (0, 1/2)$, $x \in \mathbb{C}^n$ and every integer $k \geq 1$, any $S \subseteq [n]$, $|S| = k$, if $||x||_\infty \leq R^* \cdot ||x_{[n]\setminus S}||_2/\sqrt{k}, R^* = n^{O(1)}$, an invocation of* ESTIMATE$(\hat{x}, S, k, \epsilon, R^*)$ *(Algorithm 2) returns $\chi^* \in \mathbb{C}^n$ such that*

$$||(x-\chi^*)_S||_2^2 \leq \epsilon \cdot ||x_{[n]\setminus S}||_2^2$$

using $O_\delta(\frac{1}{\epsilon}k)$ samples and $O_\delta(\frac{1}{\epsilon}k\log^{3+\delta} n)$ time with at least $4/5$ success probability.

Note that even when $x_{[n]\setminus S} \equiv 0$, at least k samples are needed to recover x_S from $\widehat{x}$, implying that our result is asymptotically optimal for constant ϵ. A linear sketch with $O(k)$ measurements and $O(k)$ recovery time that provides the guarantee in (1) was presented in [Pri11], but this solution uses general linear measurements as opposed to the more restrictive Fourier measurements. To the best of our knowledge, the estimation problem with guarantees (1) has not been studied explicitly in the setting of Fourier measurements. We now describe 'folklore' results and give a comparison with Theorem I.1.

Estimation from Fourier measurements via least squares: Recall that the problem is as follows: given a set $S \subseteq [n]$, estimate x_S from a small number of Fourier measurements of x, i.e. from a small number of accesses to $\widehat{x}$. A popular approach is to select a subset $T \subseteq [n]$ of frequencies and solve the least squares problem

$$\min_{y \in \mathbb{C}^n, \mathrm{supp}\, y \subseteq S} ||\widehat{y}_T - \widehat{x}_T||_2^2. \tag{2}$$

A natural choice is to let T be a (multi)set of frequencies selected uniformly at random with replacement from $[n]$. The solution to (2) is then provided by the normal equations $y_{OPT} = (F_{T,S}^* F_{T,S})^{-1} F_{T,S}^* x$, where $Fx \in \mathbb{C}^n$ is the Fourier transform of x, and $F_{T,S}$ is the $T \times S$ submatrix of F scaled by $\sqrt{n/|T|}$. Writing $x = x_S + x_{[n]\setminus S}$, so that $\widehat{x}_T = F_{T,S}x_S + F_{T,[n]\setminus S}x_{[n]\setminus S}$, we get $y_{OPT} = (F_{T,S}^* F_{T,S})^{-1} F_{T,S}^* \widehat{x}_T = x_S + (F_{T,S}^* F_{T,S})^{-1} F_{T,S}^* F_{T,[n]\setminus S} x_{[n]\setminus S}$, where the second term corresponds to the estimation error due to tail noise. Thus, if T is such that $\frac{1}{2} I_S \preceq F_{T,S}^* F_{T,S} \preceq 2 I_S$, then $||y_{OPT} - x||_2 = O(1) \cdot ||x_{[n]\setminus S}||_2$ with constant probability. A simple application of matrix Chernoff bounds shows that the spectral bound $\frac{1}{2} I_S \preceq F_{T,S}^* F_{T,S} \preceq 2 I_S$ is satisfied when $|T| \geq C|S|\log|S|$ for an absolute constant C. Note that the analysis above is tight, as for certain choices of $S \subseteq [n]$ at least $\Omega(|S|\log|S|)$ samples are needed even to ensure that $F_{S,T}^* F_{S,T}$ is invertible. For example, suppose that $S = (n/k) \cdot [k]$, where k divides n, so that the signal $\widehat{x}$ is k-periodic. In this case x only becomes recoverable from $\widehat{x}_T$ as long as T contains at least one element of every conjugacy class of $\mathbb{Z}_n$ modulo k, and by a Coupon Collection argument $\Omega(k \log k)$ samples are needed to ensure that this is the case. To summarize, the sample complexity of least squares with a random T is at least $\Omega(k \log k)$. Another significant disadvantage of this approach is that solving the least squares problem requires at least $\Omega(k^2)$ runtime using current techniques. Of course, given the knowledge of S one may be able to design a better than random set T, but no such construction is known for general supports S. As Theorem I.1 shows, there exists a distribution over sampling patterns T that is *oblivious to* S and allows decoding from $O(k)$ samples in $k \log^{O(1)} n$ time.

Estimation from Fourier measurements via Fourier hashing: Estimation of a subset S of coefficients of x using Fourier measurements can be performed using the idea of Fourier hashing (via filtering) commonly used in the Sparse FFT literature. In this approach one round of hashing allows one to compute estimates w_i for $x_i, i \in S$ such that

$$|w_i - x_i| \leq \alpha ||x||_2^2/k$$

using $O(k/\alpha)$ samples in Fourier domain and $O((k/\alpha)\log(k/\alpha))$ runtime. Here $\alpha \in (0,1)$ is the

oversampling parameter, which is normally set to a small constant, as it directly affects runtime and sample complexity. The approach is similar to standard hashing techniques such as CountSketch [CCFC02], but the crucial difference is that the error bound depends on the *energy of the entire signal* as opposed to energy of the tail[1]. Indeed, in general one can have $||x||_2^2 \gg n^{\Omega(1)} \cdot ||x_{[n]\setminus S}||_2^2$, meaning that one round of hashing gives results that are very far from estimating x_S up to the energy of the 'noise', i.e. elements outside S. This can be fixed by iterating the estimation process on the residual signal. A naive implementation and analysis results in $\Theta(k \log n)$ measurements (due to $\log n$ iterations of refinement) and $k \log^{O(1)} n$ time. Recent works on saving samples by reusing measurements [IK14], [Kap16] can lead to improvements over the factor $\log n$ blow up in sample complexity, but all prior approaches inherently lead to $\omega(1)$ factor loss in the number of samples, as we argue below.

Our results on recovery: The second version of the problem is the Sparse FFT (recovery) problem with ℓ_2/ℓ_2 guarantees: we are given access to $\widehat{x}$, a precision parameter $\epsilon > 0$ and a sparsity parameter k, and would like to output x' such that

$$\|x - x'\|_2^2 \leq (1+\epsilon) \min_{k\text{-sparse } y} \|x - y\|_2^2, \tag{3}$$

Note that here we are not provided with any information about the 'heavy' coefficients of x, and the hardest and most sample intensive part of the problem is to recover the identities of the 'heavy' elements.

It is known that any (randomized, non-adaptive) algorithm whose output satisfies (3) with at least constant probability must use $m = \Omega(k \log(n/k))$ samples [DIPW10]. An algorithm that matches this bound for every $k \leq n^{1-\delta}$ was recently proposed by [IK14]. The algorithm of [IK14] required $\Omega(n)$ runtime, however, leaving open the problem of achieving sample-optimality in $k \log^{O(1)} n$, or even just *sublinear time*. Sublinear time algorithms that come close to the optimal sample complexity (within an $O(\log \log n)$ factor) have been proposed [IKP14], [Kap16], but no algorithm was able to match the lower bound to within constant factors using sublinear runtime[2]. As we argue below, achieving the $O(k \log n)$ bound in sublinear time appears to require a fundamentally different approach to Fourier hashing, which we provide in this work. Our new technique results in an algorithm that matches the lower bound of [DIPW10] up to constant factors for every k polynomially bounded away from n (i.e. $k \leq n^{1-c}$ for a constant $c > 0$) in sublinear time (the runtime can be brought closer to $k \log^3 n$ by setting the parameter δ to a small constant which affect sample complexity by a constant factor):

Theorem I.2. *For any $\epsilon \in (1/n, 1), \delta \in (0, 1/2)$, $x \in \mathbb{C}^n$ and any integer $k \geq 1$, if $R^* \geq ||x||_\infty/\mu, R^* = n^{O(1)}$, $\mu^2 \geq ||x_{[n]\setminus[k]}||_2^2/k$, $\mu^2 = O(||x_{[n]\setminus[k]}||_2^2/k)$, $\text{SPARSEFFT}(\hat{x}, k, \epsilon, R^*, \mu)$ (Algorithm 3 in the full version [Kap17]) solves the ℓ_2/ℓ_2 sparse recovery problem using $O_\delta(k \log n) + O(\frac{1}{\epsilon} k \log n)$ samples and $O_\delta(\frac{1}{\epsilon} k \log^{4+\delta} n)$ time with at least $4/5$ success probability.*

We now discuss the technical difficulties that our approach overcomes. In this discussion we concentrate mainly on the estimation problem, as it is easier than sparse recovery, but at the same time exhibits all the relevant technical challenges. We first describe known sample optimal and efficient solutions that use arbitrary linear measurements, and then outline the difficulties that one faces when working with Fourier measurements.

Estimation and sparse recovery with arbitrary linear measurements: If arbitrary linear measurements are allowed, one takes, 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 a random hash function $h : [n] \to [B]$ and random signs $s_i \in \{-1, +1\}$. Since we are hashing in a number of buckets a constant factor (say, 100) larger than the sparsity of the signal, a large fraction (say, $\approx 90\%$) of the top k components are likely to be isolated in a bucket, and not have too much noise (i.e. elements other than the top k) hash into the same bucket. For such isolated elements we can approximate their value *up to the noise that hashes into the same bucket* (in the case of sparse recovery, we perform $O(\log(n/k))$ specially crafted linear measurements using the same hash function h to recover the identity of the isolated element). This lets us estimate (resp. recover) $\approx 90\%$ of the top k elements of the signal, we subtract them off and recurse on the remaining $\approx 10\%$ of the top k elements, hashing into $k/2$ buckets this time. In general, for $t = 1, 2, \ldots, O(\log k)$ we choose a random hash function $h_t : [n] \to [B_t]$, where $B_t = 100k/2^{t-1}$, say (in the case of sparse recovery we take $O(\log(n/k))$ measurements using each of these hash functions). One can show [GLPS10] that after $O(\log k)$ iterations of the hashing, recovery and subtraction process we recover an approximation to x that satisfies (1) (resp. (3) in case of recovery). The sample complexity of this process is dominated by the sample complexity of the first iteration, where we use $B_1 = 100k$ buckets, resulting in a $O(k)$ (resp. $O(k \log(n/k))$) bound on the sample complexity overall. Note that the recovery process only uses every hash function h_t once, at step t: those elements that are isolated under this hashing are perfectly recovered and essentially 'disappear' from the system, so h_t can be discarded!

A natural approach to estimation and recovery with Fourier measurements and why it fails: In order to achieve $O(k)$ (resp. $O(k \log n)$) sample complexity using Fourier

[1]One way to improve the error bound is to use strong filters [HIKP12b], but that requires a $\Omega(k \log n)$ samples.

[2]In this paper we are only interested in algorithms that work for worst case signals. If probabilistic assumptions on the signal are made, better results are possible in some settings (see, e.g. [GHI+13]).

measurements (i.e. in Sparse FFT) it seems natural to revisit the original idea used in recovery from arbitrary linear measurements that we outlined above. More precisely, we could follow the strategy of choosing, for $t = 1, 2, \ldots, O(\log k)$, a random hash function $h_t : [n] \to [B_t]$, where $B_t = 100k/2^{t-1}$. The problem is that in order to ensure that we hash into B buckets at the cost of $O(B)$ samples, we need to commit to working with rather low quality buckets implemented using crude filters (see section II) and this causes 'leakage' between hash buckets. Given this complication, it is not clear at all if estimation (resp. recovery) can be made to work: while with 'ideal' hashing each element isolated in a hashing was identified and estimated *up to amount of noise in its bucket*, here due to the leakage of our simple filters identification of nominally isolated elements can be precluded by interference from other head elements! This means that the elements that were isolated in the first hashing do not 'disappear' from the system (as they essentially do with 'ideal' hashing described above in the context of arbitrary linear measurements), but are reduced in value by only about a constant factor, and will influence the recovery process using the second hashing etc. To put this in perspective, note that when for each $t > 10$, say, we hash into $B_t = 100k/2^{t-1}$ buckets, we generally get $\Omega(k)$ original elements hashing to $\ll k$ buckets! These elements have of course been reduced in value somewhat, but not to the extent that their contribution to $B_t \approx k/2^{t-1}$ buckets is negligible.

The discussion above implies that two difficulties must be overcome to achieve $O(k)$ (resp. $O(k \log n)$) sample complexity. First, since one round of hashing can at most reduce the 'isolated' elements in the residual by a constant factor, $\Omega(\log n)$ iterations are necessary. Furthermore, the process must be set up in such a way that the $\Omega(\log n)$ iterations operate on the same hash functions, and at the same time no adversarial correlations arise to hinder the estimation process. The second difficulty is more subtle, but the harder one to deal with – this is exactly where our main contribution comes in. Note that if several levels of hashing are used, as above there could be elements whose total contribution to estimation error *over all levels* $t > 1$ *is* $\omega(1)$. Indeed, it is easy to see that some of the top k elements will participate in repeated collisions for many values of $t > 1$. Such elements could pose significant difficulties, as they introduce large errors to the identification and estimation process. This issue arises because we reuse hashings that hash $\Omega(k)$ elements into $\ll k$ buckets.

Our techniques: a new hashing scheme and isolation on average: To overcome the difficulties outlined above, we use the following approach. As above, we choose a sequence of hash functions h_t that hash the signal into a geometrically decreasing number of buckets. However, a crucial modification is that for each t we repeat the hashing process independently R_t times for an increasing sequence R_t (we use a geometrically increasing sequence, and denote the sequence of hashings by $h_{t,s}$ with $s = 1, \ldots, R_t$ for each t). As we show below in Section III, the independent repetitions ensure, at a high level, that despite the fact that most elements collide in multiple hashings, the fraction of such collisions is small, ensuring that estimation errors do not propagate – see Lemma III.1 and Remark III.2 after the lemma.

We give a formal analysis of our scheme in the rest of the paper, and provide intuition as to why our scheme fixes the problem outlined above now. Specifically, we would like to see that the head elements do not contribute a large fraction of their weight as estimation error in hashings $h_{t,s}$ for $t > 1$. The reason is that, as we show below, given the hash functions $\{h_{t,s}\}$ the set S of head elements can be partitioned into sets $S = S_1 \cup S_2 \cup \ldots \cup S_T, |S_1| \gg |S_2| \gg \ldots \gg |S_T|$ so that for every $t > 1$ every element of S collides with at least one element of S_t in *no more than* $R_t^{1-\delta}$ *out of the* R_t *hashings* $h_{t,s}$ *at iteration* t, for some constant $\delta > 0$ (choosing δ small improves runtime, at the expense of sample complexity; any small constant $\delta > 0$ leads to asymptotically sample optimal results). Thus, even though there are many collisions, **on average over** $s \in [1 : R_t]$ every element in S collides with at most $\approx R_t^{-\delta}$ elements of S_t – we refer to this property as 'isolation on average'. Since we choose the number of hashings R_t to increase geometrically, the error contributed by an element of S *over all hashings* is no more than $\sum_{t \geq 1} R_t^{-\delta} = O_\delta(R_t^{-\delta}) \ll 1$. This fact allows us to argue that iterative decoding converges (see section III). Achieving small runtime with such a scheme requires a delicate balance of parameters, which we exhibit in Section IV.

Our techniques: majorizing sequences for controlling residual signals: Lastly, one should note that the discussion above rests heavily on our ability to control the sequence of residual signals that arise throughout the update process (both in estimation and recovery). We achieve this by showing that residual signals arising during the update process are *majorized* by short (polylogarithmic length) sequence of signals (referred to as a *majorizing sequence*). We present the application to estimation here, and defer the application to recovery to the full version of the paper [Kap17].

Significance for future work: We feel that the idea of 'isolation on average' may prove useful in further developments in the area. For example, it would be interesting to see if measurement reuse using our techniques can improve sample complexity of to sublinear algorithms for *model based sparse recovery* from Fourier measurements, i.e. to Sparse FFT algorithms that *exploit structure of input signals beyond the sparsity assumption* (a sublinear time algorithm for model based Sparse FFT for the block-sparse model was recently presented in [CKSZ17]). A strong step in this direction would consist of removing the reliance of our techniques on the ℓ_1 norm of the residual signal as

the measure of progress, and introducing an approach to measurement reuse while provably reducing the ℓ_2 norm of the residual during the iterative process.

Organization: The proofs of Theorem I.1 and Theorem I.2 rely on a shared set of lemmas that enable analysis via 'isolation on average', with the main technical lemma being Lemma III.1 (see also Remark III.2 after the lemma). We present these lemmas first (Sections II and III), then prove Theorem I.1 (Section IV). The proof of Theorem I.2 is deferred to the full version [Kap17] due to space constraints.

II. Preliminaries and basic notation

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, where n is a power of 2. We use the notation $\omega = e^{2\pi i/n}$ for the root of unity of order n. The forward and inverse Fourier transforms are given by

$$\hat{x}_f = \frac{1}{\sqrt{n}} \sum_{i \in [n]} \omega^{-if} x_i \quad \text{and} \quad x_j = \frac{1}{\sqrt{n}} \sum_{f \in [n]} \omega^{jf} \hat{x}_f \qquad (4)$$

respectively, where $f, j \in [n]$. 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). We assume that entries of x are integers bounded by a polynomial in n.

A. Filters, hashing and pseudorandom permutations

We will use pseudorandom spectrum permutations, which we now define. We write $\mathcal{M}_{odd}$ for the set of odd numbers between 1 and n. For $\sigma \in \mathcal{M}_{odd}, q \in [n]$ and $i \in [n]$ let $\pi_{\sigma,q}(i) = \sigma(i-q) \mod n$. Since $\sigma \in \mathcal{M}_{odd}$, this is a permutation. Our algorithm will use π to hash heavy hitters into B buckets, where we will choose $B \approx k$. We will often omit the subscript σ, q and simply write $\pi(i)$ when σ, q is fixed or clear from context. For $i \in [n]$ we let $h(i) := \text{round}((B/n)\pi(i))$ be a hash function that maps $[n]$ to $[B]$, and for $i, j \in [n]$ we let $o_i(j) = \pi(j) - (n/B)h(i)$ be the "offset" of $j \in [n]$ relative to $i \in [n]$. We always have B a power of two.

Definition II.1. *Suppose that σ^{-1} exists $\mod n$. For $a, q \in [n]$ we define the permutation $P_{\sigma,a,q}$ by $(P_{\sigma,a,q}\hat{x})_i = \hat{x}_{\sigma(i-a)}\omega^{i\sigma q}$.*

Lemma II.2. $\mathcal{F}^{-1}(P_{\sigma,a,q}\hat{x})_{\pi_{\sigma,q}(i)} = x_i \omega^{a\sigma i}$

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

$$\mathrm{Err}_k(x) = \min_{k-\text{sparse } y} ||x-y||_2 \quad \text{and} \quad \mu^2 = \mathrm{Err}_k^2(x)/k. \qquad (5)$$

In this paper, we assume knowledge of μ (a constant factor upper bound on μ suffices). We also assume that the signal to noise ratio is bounded by a polynomial in the length n of the signal, namely that $R^* := ||x||_\infty/\mu \le n^C$ for a constant $C > 0$. It will be convenient to use the notation $\mathbb{B}_\infty(x, r)$ to denote the interval of radius r around x: $\mathbb{B}_\infty(x, r) = \{y \in [n] : |x-y|_\circ \le r\}$, where $|x-y|_\circ$ is the circular distance on $\mathbb{Z}_n$. For a real number a we write $|a|_+$ to denote the positive part of a, i.e. $|a|_+ = a$ if $a \ge 0$ and $|a|_+ = 0$ otherwise.

We will use the following

Definition II.3 (Flat filter with B buckets and sharpness F)**.** *A sequence $G \in \mathbb{R}^n$ symmetric about zero with Fourier transform $\widehat{G} \in \mathbb{R}^n$ is called a* flat filter with B buckets and sharpness F *if* **(1)** *$G_j \in [0,1]$ for all $j \in [n]$;* **(2)** *$G_j \ge 1 - \left(\frac{1}{4}\right)^{F-1}$ for all $j \in [n]$ such that $|j| \le \frac{n}{2B}$; and* **(3)** *$G_f \le \left(\frac{1}{4}\right)^{F-1}\left(\frac{n}{B|j|}\right)^{F-1}$ for all $j \in [n]$ such that $|j| \ge \frac{n}{B}$.*

We use a construction of such filters from [CKSZ17]:

Lemma II.4 ([CKSZ17], Lemma 2.1)**.** (Compactly supported flat filter with B buckets and sharpness F) *Fix the integers (n, B, F) with n a power of two, $B < n$, and $F \ge 2$ an even number. There exists an (n, B, F)-flat filter $G \in \mathbb{R}^n$, whose Fourier transform $\widehat{G}$ is supported on a length-$O(FB)$ window centered at zero in time domain.*

Note that most of the mass of the filter is concentrated in an interval 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 $i \in [n]$ one has $G^{-1}_{o_i(i)} \le 2$. We refer to the parameter F as the *sharpness* of the filter. Our hash functions are not pairwise independent, but possess an approximate pairwise independence property that still makes hashing using our filters effective. These properties are standard, and are stated in the full version of the paper [Kap17].

B. Measurements of the signal, notation for estimation error and basic bounds

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 $H = (\pi, B, F)$. Formally, 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. We give the formal definition below.

Definition II.5 (Hashing $H = (\pi, B, F)$)**.** *For a permutation $\pi = (\sigma, q)$, parameters $B > 1$ and F, a* hashing *$H := (\pi, B, F)$ is a function mapping a signal $x \in \mathbb{C}^n$ to B signals $H(x) = (u_s)_{s \in [B]}$, where $u_s \in \mathbb{C}^n$ for each $s \in [B]$, such that for each $i \in [n]$ $u_{s,i} = \sum_{j \in [n]} G_{\pi(j)-(n/B)\cdot s} x_j \omega^{i\sigma j} \in \mathbb{C}$, where G is a filter with B buckets and sharpness F constructed in Lemma II.4.*

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

Definition II.6 (Measurement $m = m(x, H, a)$)**.** *For a signal $x \in \mathbb{C}^n$, a hashing $H = (\pi, B, F)$ and a parameter $a \in [n]$, a* measurement $m = m(x, H, a) \in \mathbb{C}^B$ *is the B-dimensional complex valued vector of evaluations of a hashing $H(x)$ at a point $a \in [n]$, i.e. for $s \in [B]$ $m_s = \sum_{j\in[n]} G_{\pi(j)-(n/B)\cdot s} x_j \omega^{a\sigma j}$, where G is a filter with B buckets and sharpness F constructed in Lemma II.4.*

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 $a \in [n]$, i.e. computes the measurement $m(x, H, a)$ (the computation is done with polynomial precision). We will use the following lemma, which is rather standard:

Lemma II.7. HASHTOBINS*$(\hat{x}, \chi, (H, a))$, where $H = (\pi, B, F)$, computes $u \in \mathbb{C}^B$ such that for any $i \in [n]$, $u_{h(i)} = \Delta_{h(i)} + \sum_j G_{o_i(j)}(x - \chi)_j \omega^{a\sigma j}$, where G is the filter defined in section II, and for all $i \in [n]$ we have that $\Delta^2_{h(i)} \le \|\chi\|_2^2 \cdot n^{-c}$ is a negligible error term (and $c > 0$ is an absolute constant that governs the precision that semi-equispaced FFT is invoked with). It takes $O(BF)$ samples, and $O(F \cdot B \log B + \|\chi\|_0 \log n)$ time.*

We now introduce relevant notation for bounding the error induced by our measurements in locating or estimating an element $i \in [n]$. For a hashing $H = (\pi, B, F)$ and an evaluation point $z \in [n]$, we have by Definition II.6

$$m_{h(i)}(x, H, z) = \sum_{j\in[n]} G_{o_i(j)} x_j \omega^{z\sigma j},$$

where the filter $G_{o_i(j)}$ is the filter corresponding to hashing H (note that $o_i(j)$ implicitly depends on π). In particular, one has:

$$G^{-1}_{o_i(i)} m_{h(i)} \omega^{-z\sigma i} = x_i + \underbrace{G^{-1}_{o_i(i)} \sum_{j\in[n]\setminus\{i\}} G_{o_i(j)} x_j \omega^{z\sigma(j-i)}}_{\text{noise term}}$$

A common idea underlying our analysis of estimation and recovery is to split the estimation/recovery error induced on an element i into the contribution from the carefully defined 'head' of the signal and the contribution from the 'tail'. The 'head' of the signal is denoted by a set $S \subseteq [n]$ throughout the paper. For each $i \in [n]$ we write

$$G^{-1}_{o_i(i)} m_{h(i)} \omega^{-z\sigma i} = x_i + \underbrace{G^{-1}_{o_i(i)} \cdot \sum_{j\in S\setminus\{i\}} G_{o_i(j)} x_j \omega^{z\sigma(j-i)}}_{\text{noise from 'heavy' elements}} + \underbrace{G^{-1}_{o_i(i)} \cdot \sum_{j\in[n]\setminus(S\cup\{i\})} G_{o_i(j)} x_j \omega^{z\sigma(j-i)}}_{\text{'tail' noise}} \tag{6}$$

We now define special notation for the two noise terms in (6). These two noise terms will be handled very differently in our analysis.

Noise from heavy hitters: The first term in (6) corresponds to noise from $x_{S\setminus\{i\}}$, i.e. noise from 'head' of the signal. For every $i \in S$, hashing H we let

$$e^{head}_i(H, x) := G^{-1}_{o_i(i)} \cdot \sum_{j\in S\setminus\{i\}} G_{o_i(j)} |x_j|. \tag{7}$$

We thus get that $e^{head}_i(H, x)$ upper bounds the absolute value of the first error term in (6) *for every value of evaluation point z* (note that $e^{head}(H, x)$ only depends on the hashing H and x). Note that $G \ge 0$ by Lemma II.4 and Definition II.3 as long as F is even, which is the setting that we are in. We will often use several hashings to estimate or locate an element i. It is thus convenient to define, for a sequence of hashings $H_1, \ldots, H_r$

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

where for a list of reals $u_1, \ldots, u_s$ and a number $f \in (0, 1)$ we let 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 $i \in [n], z \in [n]$, permutation $\pi = (\sigma, q)$ and hashing $H = (\pi, B, F)$

$$e^{tail}_i(H, z, x) := \left| G^{-1}_{o_i(i)} \cdot \sum_{j\in[n]\setminus(S\cup\{i\})} G_{o_i(j)} x_j \omega^{z\sigma(j-i)} \right|. \tag{9}$$

With this definition in place $e^{tail}_i(H, z, x)$ upper bounds the absolute value of second term in (6). We will sometimes use several hashings and values z to obtain better estimates. For a sequence $\{(H_r, z_r)\}_{r=1}^{r_{max}}$ for some $r_{max} \ge 1$ we let

$$e^{tail}_i(\{H_r, z_r\}, x) := \text{quant}^{1/5}_r \left| G^{-1}_{o_i(i)} \cdot \sum_{j\in[n]\setminus(S\cup\{i\})} G_{o_i(j)} x_j \omega^{z\sigma(j-i)} \right|, \tag{10}$$

where $o_i(j)$ on the rhs implicitly depends on the hashing H.

The definitions above are sufficient for our analysis of the signal estimation procedure in Section IV. With the definitions above we can state the following simple guarantees on the performance of a basic estimation procedure that is the main building block of our analysis of the more powerful ESTIMATE primitive in Section IV.

Lemma II.8 (Bounds on estimation quality for ESTIMATEVALUES)**.** *For every $x, \chi \in \mathbb{C}^n$, every $L \subseteq [n]$, every set $S \subseteq [n]$ the following conditions hold for functions e^{head} and e^{tail} defined with respect to S (see* (7) *and* (9)*). If r_{max}*

is larger than an absolute constant, then for every sequence $H_r = (\pi_r, B, F), r = 1, \ldots, r_{max}$ *of hashings and every sequence* $a_1, \ldots, r_{max}$ *of evaluation points the output* w *of*

$$\text{EstimateValues}(\chi, L, \{(H_r, a_r, m(x, H_r, a_r))\}_{r=1}^{r_{max}})$$

satisfies, for each $i \in L$

$$|w_i - (x-\chi)_i| \leq 2 \cdot \mathit{quant}_r^{1/5} e_i^{head}(H_r, x-\chi) + 2 \cdot \mathit{quant}_r^{1/5} e_i^{tail}(H_r, a_r, x-\chi) + n^{-\Omega(c)},$$

where $c \geq 2$ *is an absolute constant that governs the precision of our approximate semi-equispaced FFT computations (see* HashToBins*, Lemma II.7). The sample complexity is bounded by* $O(FBr_{max})$*, and the runtime by* $O((F \cdot B \cdot \log n + ||\chi||_0 \log n + |L|) \cdot r_{max})$.

The proof of the lemma is given in the full version [Kap17]. The proof is rather standard modulo our definitions of e^{head} and e^{tail}, as well as the fact that the statement of the lemma is entirely deterministic. We will later apply this lemma to random hashings and evaluation points, but the deterministic nature of the claim will be crucial in analyzing measurement reuse.

As both our Estimate (Section IV) and SparseFFT (presented in the full version [Kap17]) algorithms iteratively update the signal, we will need to analyse the performance of EstimateValues on various residual signals derived from the original input signal x. The notion of a *majorant* is central to this part of our analysis:

Definition II.9 (Majorant)**.** *For any* $S \subseteq [n]$ *and any* $x, y \in \mathbb{C}^n$ *we say that* y *is an majorant for* x *with respect to* S *if* $|x_i| \leq |y_i|$ *for all* $i \in S$.

With this definition and definition of e^{head} above the following crucial lemma follows immediately:

Lemma II.10. *For every hashing* H*, every set* $S \subseteq [n]$ *one has for every pair* $x, y \in \mathbb{C}^n$ *that if* $x \prec_S y$*, then for every* $i \in [n]$ $e_i^{head}(H, x) \leq e_i^{head}(H, y)$.

Proof: Recall that by (7) one has $e_i^{head}(H, x) = G^{-1}_{o_i(i)} \cdot \sum_{j \in S \setminus \{i\}} G_{o_i(j)} |x_j|$. Since $G \geq 0$ by Definition II.3 and Lemma II.4, we have by using $x \prec_S y$ that $e_i^{head}(H, x) = G^{-1}_{o_i(i)} \cdot \sum_{j \in S \setminus \{i\}} G_{o_i(j)} |x_j| \leq G^{-1}_{o_i(i)} \cdot \sum_{j \in S \setminus \{i\}} G_{o_i(j)} |y_j| = e_i^{head}(H, y)$ as required. ∎

III. Isolating partitions

In this section we prove the main lemmas that allow us to reason about performance of the SNR reduction process in our algorithms (Algorithm 2 and Algorithm 3 in the full version [Kap17]). Both algorithms perform SNR reduction (see lines 16 to 22 in Algorithm 2) using two loops. The first loop (over r), controls the ℓ_1 norm of the signal, with the (upper bound on the) norm being reduced by a factor of 4 in each iteration. This reduction is achieved via a sequence of calls to EstimateValues (in Estimate, Algorithm 2) using a separate collection of hashings for each $t = 1, \ldots, T$. Our correctness analysis for this process proceeds by showing that, with high constant probability over the choice of the hashings $\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$ any set S of size $\approx k$ the hashings induce a partition of S into at most T sets $S_1 \cup \ldots \cup S_T$ such that hashings used in the t-th round allow the algorithm to make progress on elements in S_t. The formalization of this claim is given by the following lemma, which is central to our analysis:

Lemma III.1. *For every integer* $k \geq 1$*, every* $S \subseteq [n], |S| \leq k$*, every* $\delta \in (0, 1/2)$*, if the parameters* B_t, R_t *are selected to satisfy* **(p1)** $R_t = C_1 \cdot 2^t$ *and* **(p2)** $B_t \geq C_2 \cdot k / R_t^2$ *for every* $t \in [0 : T]$*, where* C_1 *is a sufficiently large constant and* C_2 *is sufficiently large as a function of* C_1 *and* δ*, then the following conditions hold.*

For every collection of hashings $\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$*, if the filters used in hashings* $H_{t,s}$ *are at* F*-sharp for even* $F \geq 6$ *for every* $t \in [1 : T]$*, and* S *admits a* δ*-isolating partition (as per Definition III.7)* $S = S_1 \cup \ldots \cup S_T$ *with respect to* $\{H_{t,s}\}$*, then for every* $x, \chi \in \mathbb{C}^n, x' = x - \chi$*, for every* $t \in [1 : T]$ *one has* $||e_{S_t}^{head}(\{H_{t,s}\}_{s \in [1:R_t]}, x')||_1 \leq 20 R_t^{-\delta} ||x'_S||_1$.

The proof of the lemma is deferred to the full version [Kap17] due to space constraints.

Remark III.2. *Note that the result of Lemma III.1 implies that the cumulative error induced by the entire set* S *of 'heavy' coefficients on* S_t *is only a* $\approx R_t^{-\delta}$ *fraction of the* ℓ_1 *norm of* x'_S*, despite the fact that when estimating* S_t *we hash into* $B_t \ll k$ *buckets, and in general each bucket will contain many elements of* S*. Furthermore, if we choose* R_t *to increase fast enough so that* $\sum_{t \geq 1} R_t^{-\delta} \ll 1$*, we get that the cumulative contribution of elements in* S *to estimation/location error over all* $t \geq 1$ *is less than* 1*, meaning that errors do not accumulate much. This is exactly what we achieve by setting* $R_t = C_1 2^t$ *for a large constant* $C_1 > 1$ *– see proof of Lemma IV.1 in Section IV.*

In the rest of the section we first introduce relevant notation and in particular define the central notion of an *isolating partition* of the set S of head elements (in section III-A), then prove that any fixed set S of size about k admits an isolating partition with at least high constant probability over the choice of hashings $\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$ (section III-B), and show how to construct the partition efficiently (Lemma III.9) if the set S is given explicitly (used in Algorithm 2, line 13). Using this result we then give a proof of Lemma III.1.

A. Main definitions

Let S be any subset of $[n]$ (we will later instantiate S to the set of 'large' elements of x). We now define a decomposition of the set S into $T = \frac{1}{1-\delta} \log_2 \log(k+1) + O(1)$ disjoint sets $S_1, S_2, \ldots, S_T$ with respect to a

sequence $1 \leq R_0 \leq R_1 \leq R_2 \leq \ldots \leq R_T$ and hashings $\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$. We start with several auxiliary definitions.

Definition III.3 (t-Collision)**.** *We say that an element $a \in [n]$ participates in a t-collision with another element $b \in [n]$ under hashing $H = (\pi, B, F)$ if a hashes within at most t buckets of b under H, i.e. if $|\pi(a) - \pi(b)| \leq \frac{n}{B}(t-1)$.*

Definition III.4 (δ-bad element)**.** *For $\delta \in (0,1)$, for each $t \in [1:T]$, sequence $1 \leq R_0 \leq R_1 \leq \ldots \leq R_T$, where $T \geq 1$ is an integer, we say that an element a of S is δ-*bad for *S_t with respect to a partition $S = S_1 \cup S_2 \cup \ldots \cup S_T$ and hashings $\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$ if a participates in an R_t-collision with at least one element of S_t under more than a $R_t^{-\delta}$ fraction of hashings $H_{t,1}, \ldots, H_{t,R_t}$.*

Definition III.5 (λ-crowded element)**.** *For a hashing $H = (\pi, B, F)$ and a real number $\lambda \in (0,1)$, an element $a \in [n]$ is λ-crowded at scale $q \geq 0$ by a set $Q \subseteq [n]$ if $\left|\mathbb{B}\left(\pi(a), \frac{n}{B} \cdot 2^q\right) \cap \pi(Q \setminus \{a\})\right| \geq \lambda 2^{2q}$. We say that an element a is simply λ-crowded if it is λ-crowded at least at one scale $q \geq 0$.*

Remark III.6. *The intuition for the definition above is that if the permutation π was pairwise independent, then for every $a \in [n]$ the expectation of $\left|\mathbb{B}\left(\pi(a), \frac{n}{B} \cdot 2^q\right) \cap \pi(Q \setminus \{a\})\right|$ would be about 2^q if $|Q| \leq B$, i.e. about the number of buckets that fall into the interval. We say that an element is λ-crowded when the number of elements in its vicinity exceeds $\lambda 2^{2q}$ for at least one scale q, i.e. exceeds expectation by a $\lambda 2^q$ factor. Our choice of $\lambda = R_t^{-3}$ serves the purpose of enforcing that no element of S_t is hashed too close to another element of S_t, and no (large) neighborhood of any element of S_t it too crowded. These two parameter regimes have somewhat distinct applications in the proof of Lemma III.1 – see footnotes 3 and 4 in the proof of the lemma on pages 13 and 14 respectively.*

Definition III.7 (δ-isolating partition)**.** *For $\delta \in (0,1)$, for any $k \geq 1$ and any $S \subseteq [n], |S| \leq k$, a partition $S = S_1 \cup S_2 \cup \ldots \cup S_T$ of $S \subseteq [n]$ into disjoint subsets is δ-*isolating *with respect to a sequence of integers $1 \leq R_0 \leq R_1 \leq \ldots \leq R_T$ and hashings $\{H_{t,s}\}_{s=1}^{R_t}$ for $t = [1:T]$ if the following conditions are satisfied for each $t \in [1:T]$:*

(1) *$|S_t| \leq k \cdot \frac{R_0}{R_{t-1}} 2^{-2^{(1-\delta)\cdot(t-1)}+1}$ (the sizes of S_t's decay doubly exponentially);*

(2) *no element of S_t is R_t^{-3}-crowded by S_t under any of $\{H_{t,s}\}_{s=1}^{R_t}$ (elements of S_t are rather uniformly spread under hashings $H_{t,s}$);*

(3) *no element of S_t R_t-collides with an element of S that is δ-bad for S_t under any of $\{H_{t,s}\}_{s=1}^{R_t}$ (collisions between S and S_t are rare).*

Note that the bound on the size of S_1 is at most k, which will be trivial for our instantiation of S. Nontrivial decay of the S_t starts at $t = 2$. Also note that property **(3)** is exactly why we call our approach 'isolation on average': as the proof of Lemma III.1 below shows, the fact that no element of S that is δ-bad for S_t collides with S_t under any of the hashings implies that every element of S has a limited contribution to estimation error, and errors do not propagate.

B. Existence of isolating partitions of S

In this section we prove that any set S of size at most k admits an isolating partition with respect to a random set of hashings as in Algorithm 2 with at least high constant probability. We prove this claim by giving an algorithm that we show constructs such a partition successfully with high constant probability. The algorithm is Algorithm 1, presented below. We prove that Algorithm 1 terminates correctly in Lemma III.8 below, assuming that the number of hashings at each step t and their parameters are chosen appropriately.

If the set S is known explicitly, the partition can be constructed efficiently by running Algorithm 1 – the details are given in Lemma III.9. An efficient construction is needed in our sample efficient primitive (see Algorithm 2). Our sample-optimal Sparse FFT algorithm is oblivious to the actual partition, as its task is to identify the set S. However, as we show in the full version [Kap17], the existence of an isolating partition of S is sufficient for the algorithm to work.

Algorithm 1 Construction of an isolating partition $\{S_j\}$

procedure CONSTRUCTPARTITION($\{\{H_{t,s}\}_{s=1}^{R_t}\}_{t=1}^{T}$)
  $S_1^1 \leftarrow S$, $t \leftarrow 1$
  while $S_t^t \neq \emptyset$ **do**
    $\text{Bad}_t \leftarrow$ {elements of S that are δ-bad wrt S_t^t under $\{H_{t,s}\}_{s\in[1:R_t]}$}
    $U_t \leftarrow$ {elements $a \in S_t^t$ that R_t-collide with Bad_t under at least one of $\{H_{t,s}\}_{s\in[1:R_t]}$}
    $V_t \leftarrow$ {elements $a \in S_t^t$ that are R_t^{-3}-crowded by S_t^t under at least one of $\{H_{t,s}\}_{s\in[1:R_t]}$}
    Set $S_{t+1}^{t+1} \leftarrow \text{Bad}_t \cup U_t \cup V_t$
    and
    $S_j^{t+1} \leftarrow S_j^t \setminus S_{t+1}^{t+1}$ for $j = 1, \ldots, t$
    $t \leftarrow t+1$
  end while
  return the partition $\{S_j^t\}_{j=1}^{t}$
end procedure

We now argue that Algorithm 1, with appropriately set parameters, constructs an isolating partition of any set $S \subseteq [n]$ that satisifies $|S| \leq k$ with at least high constant probability:

Lemma III.8. *For every integer $k \geq 1$, every $S \subseteq [n], |S| \leq k$, every $\delta \in (0, 1/2)$, if the parameters B_t, R_t are selected to satisfy* **(p1)** *$R_t = C_1 \cdot 2^t$ and* **(p2)** *$B_t \geq C_2 \cdot k/R_t^2$ for every $t \in [0:T]$, where C_1 is a sufficiently large constant and C_2 is sufficiently large as a function of C_1 and δ, then the following conditions hold.*

With probability at least $1 - 1/25$ *over the choice of hashings* $\{\{H_{t,s}\}_{s\in[1:R_t]}\}_{t=1}^{T}$ *Algorithm 1 terminates in* $T = \frac{1}{1-\delta}\log_2\log(k+1) + O(1)$ *steps. When the algorithm terminates, the output partition* $\{S_j\}_{j=1}^{T}$ *is* δ*-isolating as per Definition III.7.*

If the set S is known explicitly, Algorithm 1 admits a simple efficient implementation (the proof is given in the full version [Kap17]):

Lemma III.9. *For any integer* $k \geq 1$*, any* $S \subseteq [n], |S| \leq k$*, if the hashings* $\{\{H_{t,s}\}_{s\in[1:R_t]}\}_{t=1}^{T}$ *are such that the partition* $\{S_j\}_{j=1}^{T}$ *defined by Algorithm 1 is isolating as per Definition III.7, this partition can be constructed explicitly in time* $O\left((\sum_{t=1}^{T} R_t)\cdot|S|\log|S|\right)$.

IV. Sample efficient estimation

In this section we state our sample optimal estimation algorithm (Algorithm 2) and provide its analysis.

A. Algorithm and overview of analysis

Our algorithm (Algorithm 2) contains three major components: it starts by taking measurements m of the signal x (accessing the signal in Fourier domain, i.e. accessing $\widehat{x}$), then uses these measurements to perform a sequence of ℓ_1 norm reduction steps, reducing ℓ_1 norm of the residual signal on the target set S of coefficients to about the noise level. Finally, a simple cleanup procedure is run to convert the ℓ_1 norm bounds on the residual to ℓ_2/ℓ_2 guarantees of (1). In this section we will use the functions e^{head}, e^{tail} (see Section II) defined with respect to the set S.

Measuring $\widehat{x}$. All measurements that the algorithm takes are taken in lines 6-12, and then line 31. The measurements in lines 6-12 are taken over $T = \frac{1}{1-\delta}\log_2\log(k+1) + O(1)$ rounds for small constant $\delta \in (0, 1/2)$, where in round $t = 1, \ldots, T$ we are hashing the signal into $B_t \approx k/R_t^2$ buckets, where R_t grows exponentially with t. For each t we perform R_t independent hashing experiments of this type. This matches the setup of Lemma III.1, which is our main analysis tool (see proof of Lemma IV.1).

ℓ_1 norm reduction loop. Once the samples have been taken, Algorithm 2 proceeds to the ℓ_1 norm reduction loop (lines 16-22). The objective of this loop is to reduce the ℓ_1 norm of the residual signal on the target set S of coefficients that we would like to estimate to about the noise level, namely to $O(||x_{[n]\setminus S}||_2\sqrt{k})$. The formal guarantees are provided by

Lemma IV.1. *For every* $\delta \in (0, 1/2)$*, if* C_1, C_2 *(parameters in Algorithm 2) are sufficiently large constants, then the following conditions hold.*

For every $x \in \mathbb{C}^n$*, every integer* $k \geq 1$*, every* $S \subseteq [n]$*,* $|S| = k$*, if* $||x||_\infty \leq R^* \cdot ||x_{[n]\setminus S}||_2/\sqrt{k}, R^* = n^{O(1)}$*, the vector* $\tilde{\chi}$ *computed in line 23 of an invocation of* ESTIMATE$(\hat{x}, S, k, \epsilon, R^*)$ *(Algorithm 2) satisfies*

$$||(x-\tilde{\chi})_S||_1 \leq O_\delta(||x_{[n]\setminus S}||_2 \cdot \sqrt{k})$$

conditioned on an event $\mathcal{E}_{maj}$ *that occurs with probability at least* $1 - 2/25$.

Cleanup phase and final result. Once the ℓ_1 norm of the residual on S has been reduced to $O(||x_{[n]\setminus S}||_2\sqrt{k})$, we run the ESTIMATEVALUES procedure once to convert ℓ_1 norm bounds on the residual into ℓ_2/ℓ_2 guarantees (1). This results in a proof of Theorem I.1.

B. Proof of Lemma IV.1

We given an outline of the proof, and defer the details to the full version [Kap17].

Proof of Lemma IV.1: Recall that in this section we use the quantities e^{head} and e^{tail} defined with respect to the set S. We will also use an isolating partition of S, denoted by $S = S_1 \cup S_2 \cup \ldots \cup S_T$. We argue the existence of such a partition with high probability later.

We prove by induction on the pair (r, t) that conditional on a high probability success event $\mathcal{E}_{maj}$ (defined below) the residual signals $x - \chi^{(r,t)}$ are *majorized* on S (in the sense of Definition II.9) by a fixed sequence $y^{(r,t)}$ whose ℓ_1 norm converges to $O(||x_{[n]\setminus S}||_2 \cdot \sqrt{k})$ after $O(\log R^*)$ iterations. Since we only update elements in S, this gives the result. We now give the details of the argument. In what follows we let $\mu^2 := ||x_{[n]\setminus S}||_2^2/k$ for convenience. Note, however, that Algorithm 2 is oblivious to the value of μ: we only need an upper bound on $\log R^*$.

We start by defining the majorizing sequence $y^{(r,t)}$. We first let $y_i^{(0,0)} = R^*\mu$ for all $i \in S$ and $y_i^{(0,0)} = x_i$ otherwise. Note that $y^{(0,0)}$ trivially majorizes x as $||x||_\infty \leq R^* \cdot \mu$ by assumption of the lemma. The construction of $y^{(r,t)}$ proceeds by induction on (t, r). Given $y^{(r',t')}$, as per Algorithm 2 the next signal to be defined is $y^{(r,t)}$ with $(r, t) = (r', t'+1)$ if $t < T$ and $(r, t) = (r'+1, 1)$ otherwise (as per lines 15-22 of Algorithm 2). We now define the signal $y^{(r,t)}$ by letting for each $i \in [n]$ (recall that S_t is the t-th set in an isolating partition $S = S_1 \cup S_2 \cup \ldots \cup S_T$)

$$y_i^{(r,t)} := \begin{cases} 20e_i^{head}(\{H_{t,s}\}_{s\in[1:R_t]}, y^{(r',t')}) \\ +20e_i^{tail}(\{H_{t,s}, a_{t,s}\}_{s\in[1:R_t]}, x) \\ +n^{-\Omega(c)} & \text{if } i \in S_t \\ y_i^{(r',t')} & \text{o.w.} \end{cases} \tag{11}$$

Here $n^{-\Omega(c)}$ corresponds to the (negligible) error term due to polynomial precision of our computations. Note that there are two contributions to $y^{(r,t)}$: one coming from the previous signal in the majorizing sequence, namely $y^{(r',t')}$, and the other coming from the tail of the signal x.

We now prove by induction on (t, r) that the loop in our estimation primitive reduces the ℓ_1 norm of the residual to $O(\mu \cdot k)$ (recall that $\mu^2 = ||x_{[n]\setminus S}||_2^2/k$). Specifically, we prove that there exists an event $\mathcal{E}_{maj}$

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

1: **procedure** ESTIMATE($\hat{x}, S, k, \epsilon, R^*$)
2: $T \leftarrow \frac{1}{1-\delta}\log_2\log(k+1)+O(1)$ for a small constant $\delta \in (0, 1/2)$
3: $R_t \leftarrow C_1 \cdot 2^t$ for $t \in [1:T]$ ▷ Constant $C_1 > 0$
4: $B_t \leftarrow C_2 \cdot k/R_t^2$ for $t \in [1:T]$ ▷ Const. $C_2 > 0$
5: $G_t \leftarrow$ filter with B_t buckets and sharpness $F = 8$.
6: **for** $t = 1$ to T **do** ▷ Take samples
7: **for** $s = 1$ to R_t **do**
8: Choose $\sigma \in \mathcal{M}_{odd}$, $q \in [n]$ u.a.r., let $\pi_{t,s} \leftarrow (\sigma, q)$, $H_{t,s} := (\pi_{t,s}, B_t, F)$
9: Let $a_{t,s} \leftarrow$ an element of $[n]$ u.a.r.
10: $m(x, H_{t,s}, a_{t,s}) \leftarrow$ HASHTOBINS($\hat{x}, 0, (H_{t,s}, a_{t,s})$)
11: **end for**
12: **end for**
13: Explicitly construct a δ-isolating partition $S = S_1 \cup S_2 \ldots \cup S_T$ ▷ As per Lemma III.9
14: $\chi^{(0,0)} \leftarrow 0$
15: $r' \leftarrow 0, t' \leftarrow 0$
16: **for** $r = 0, 1, \ldots, C\log_4 R^*$ **do**
17: **for** $t = 1$ to T **do**
18: $\chi' \leftarrow$ ESTIMATEVALUES($\chi^{(r',t')}, S_t, \{(H_{t,s}, a_{t,s}, m(x, H_{t,s}, a_{t,s}))\}_{s=1}^{R_t}$)
19: $\chi^{(r,t)} \leftarrow \chi^{(r',t')} + \chi'$
20: $r' \leftarrow r$, $t' \leftarrow t$
21: **end for**
22: **end for**
23: $\tilde{\chi} \leftarrow \chi^{(C\log_4 R^*, T)}$
24: $B \leftarrow C_2 \cdot k/\epsilon$
25: $G \leftarrow$ filter with B buckets and sharpness $F = 8$.
26: $r_{max} \leftarrow O(1)$
27: **for** $r = 1$ to $O(1)$ **do**
28: Choose $\sigma_r \in \mathcal{M}_{odd}$, $q_r, a_r \in [n]$ u.a.r., let $\pi_r \leftarrow (\sigma_r, q_r)$, $H_r := (\pi_r, B, F)$
29: $m(x, H_r, a_r) \leftarrow$ HASHTOBINS($\hat{x}, \tilde{\chi}, H_r, a_r$)
30: **end for**
31: $\chi'' \leftarrow$ ESTIMATEVALUES($\tilde{\chi}, S, \{(H_r, a_r, m(x, H_r, a_r))\}_{r=1}^{r_{max}}$)
32: $\chi^* \leftarrow \tilde{\chi} + \chi''$
33: **return** χ^*
34: **end procedure**

with $\mathbf{Pr}_{\{\{H_{t,s}\}_{s\in[1:R_t]}\}_{t=1}^{T}}[\mathcal{E}_{maj}] \geq 1 - 2/25$ such that conditioned on $\mathcal{E}_{maj}$ the set S admits an isolating partition $S = S_1 \cup S_2 \cup \ldots \cup S_T$ with respect to $\{\{H_{t,s}\}\}$, and for every $(r,t) \in ([0:+\infty) \times [1:T]) \cup \{(0,0)\}$

(A) for all $q \in [1:t]$ one has $||y_{S_q}^{(r,t)}||_1 \leq (R^* \cdot (1/4)^{r+1}\mu + 2\mu) \cdot k \cdot (R_0/R_{q-1})^{\delta}$;
(B) for all $q \in [t+1:T]$ one has $||y_{S_q}^{(r,t)}||_1 \leq (R^* \cdot (1/4)^{r}\mu + 2\mu) \cdot k \cdot (R_0/R_{q-1})^{\delta}$;
(C) $||y_S^{(r,t)}||_1 \leq (2/\delta) \cdot (R^*(1/4)^r\mu + 2\mu) \cdot k$;
(D) $(x - \chi^{(r,t)}) \prec_S y^{(r,t)}$ and $\operatorname{supp}\chi^{(r,t)} \subseteq S$.

The details of proof is deferred to the full version of the paper [Kap17] due to space constraints. □

C. Proof of Theorem I.1

The proof of Theorem I.1 relies on the following simple lemma, proved in the full version [Kap17] of the paper:

Lemma IV.2. *For every $x \in \mathbb{C}^n$, every $S \subseteq [n]$, every $i \in [n]$, every integer r_{max} larger than an absolute constant, integers B, F with B a power of two and $F \geq 2$, the following conditions are satisfied for a sequence of random hashings $H_r = (\pi_r, B, F)$, and random evaluation points a_r, $r = 1, 2, \ldots, r_{max}$.*

If $Z^{head} := e_i^{head}(\{H_r\}, x) = quant_r^{1/5} e_i^{head}(H_r, x)$ (as per (8)) and $Z^{tail} := e_i^{tail}(\{H_r, a_r\}, x) = quant_r^{1/5} e_i^{tail}(H_r, a_r, x)$ (as per (10)), where e^{head} and e^{tail} are defined with respect to the set S, one has

(1) $\mathbf{E}_{\{H_r\}}\left[(Z^{head})^2\right] = O\left(\left(\frac{1}{B}||x_S||_1\right)^2\right)$;
(2) $\mathbf{E}_{\{H_r, a_r\}}\left[(Z^{tail})^2\right] = O(||x_{[n]\setminus S}||_2^2/B)$;
(3) $\mathbf{Pr}_{\{H_r\}}\left[Z^{head} > O\left(\frac{1}{B}||x_S||_1\right)\right] = 2^{-\Omega(r_{max})}$;
(4) $\mathbf{Pr}_{\{H_r\}}\left[Z^{tail} > O(||x_{[n]\setminus S}||_2/\sqrt{B})\right] = 2^{-\Omega(r_{max})}$.

Proof of Theorem I.1: Recall that we use the quantities e^{head} and e^{tail} defined with respect to the set S. Fix δ. By Lemma IV.1 we have that conditioned on a high probability event $\mathcal{E}_{maj}$ (which occurs with probability at least $1-2/25$) the vector $\tilde{\chi}$ computed in line 23 satisfies

$$||(x - \tilde{\chi})_S||_1 = O(||x_{[n]\setminus S}||_2\sqrt{k}). \tag{12}$$

To complete the proof, we show that the output χ'' of the invocation of ESTIMATEVALUES in line 31, when added to $\tilde{\chi}$, yields guarantee claimed by the lemma. First, by Lemma II.8 with S one has for each $i \in S$

$$\begin{aligned}|\chi_i'' - (x - \tilde{\chi})_i| \leq\ & 2 \cdot \text{quant}_r^{1/5} e_i^{head}(H_r, x - \tilde{\chi}) \\ & + 2 \cdot \text{quant}_r^{1/5} e_i^{tail}(H_r, a, x) \\ & + n^{-\Omega(c)},\end{aligned} \tag{13}$$

since $\operatorname{supp}\tilde{\chi} \subseteq S$.

Squaring both sides of (13), using the bound $(a+b)^2 \leq 2a^2 + 2b^2$ and taking expectations over the randomness in measurements taken in lines 27-30, we get

$$\begin{aligned}\mathbf{E}[|\chi_i'' - (x - \tilde{\chi})_i|^2] \leq\ & 8 \cdot \mathbf{E}\left[(\text{quant}_r^{1/5} e_i^{head}(H_r, x - \tilde{\chi})^2\right] \\ & + 8 \cdot \mathbf{E}\left[(\text{quant}_r^{1/5} e_i^{tail}(H_r, a_r, x))^2\right] \\ & + n^{-\Omega(c)}.\end{aligned} \tag{14}$$

We now upper bound the expectation of (14). By Lemma IV.2, **(1)** one has, letting $Z^{head} :=$

$\text{quant}_r^{1/5} e_i^{head}(H_r, x - \tilde{\chi})$ to simplify notation,

$$\begin{aligned}\mathbf{E}\left[(Z^{head})^2\right] &= O\left(\left(\frac{1}{B}||(x-\tilde{\chi})_S||_1\right)^2\right)\\ &= O\left(\left(\frac{1}{C_2 k/\epsilon}||(x-\tilde{\chi})_S||_1\right)^2\right)\\ &= O(\epsilon^2||x_{[n]\setminus S}||_2^2/(C_2 k)),\end{aligned}$$

where we used that by conditioning on $\mathcal{E}_{maj}$ one has $||(x-\tilde{\chi})_S||_1 = O(||x_{[n]\setminus S}||_2\sqrt{k})$ (by (12)).

By Lemma IV.2, **(2)** with S one has, letting $Z^{tail} := \text{quant}_r^{1/5} e_i^{tail}(H_r, a_r, x)$ to simplify notation,

$$\begin{aligned}\mathbf{E}\left[(Z^{tail})^2\right] &= O(||(x-\tilde{\chi})_{[n]\setminus S}||_2^2/B)\\ &= O(\epsilon||x_{[n]\setminus S}||_2^2/(C_2 k)),\end{aligned}$$

where we used the fact that $\operatorname{supp}\tilde{\chi} \subseteq S$.

Substituting these bounds into (14) and summing over all $i \in S$, we get

$$\begin{aligned}\mathbf{E}[||(x-\tilde{\chi}-\chi'')_S||^2] &\le O(\epsilon^2||x_{[n]\setminus S}||_2^2/C_2)\\ &\quad + O(\epsilon||x_{[n]\setminus S}||_2^2/C_2)\\ &\le (\epsilon/1000)||x_{[n]\setminus S}||_2^2\end{aligned}$$

as long as C_2 is sufficiently large.

An application of Markov's inequality then gives $||(x-\tilde{\chi}-\chi'')_S||^2 \le \epsilon||x_{[n]\setminus S}||_2^2$ with probability at least $1 - 1/1000$. By a union bound over this failure event and $\bar{\mathcal{E}}_{maj}$, we conclude that the algorithm outputs the correct answer with probability at least $1 - 3/25 \ge 4/5$.

We now upper bound the sample complexity and runtime.

Sample complexity. The sample complexity of lines 6-11 is bounded by $\sum_{t=1}^{T}\sum_{s=1}^{R_t} O(F \cdot B_t) = \sum_{t=1}^{T} R_t \cdot O(F \cdot k/R_t^2) = O(k)\cdot\sum_{t=1}^{T} 1/R_t = O(k)$ by the choice of R_t as geometrically increasing. The sample complexity of lines 27-30 is upper bounded by $O(F \cdot B) = O(k/\epsilon)$ by Lemma II.8 and choice of $F = O(1)$.

Runtime. The runtime of HASHTOBINS in line 10 of Algorithm 2 is $O(F \cdot B_t \log B_t) = O(B_t \log B_t)$ by Lemma II.7, the setting of $F = O(1)$ and the fact that the residual signal passed to the call is zero. Since this line is executed for $t = 1, \ldots, T$ and $s = 1, \ldots, R_t$, the total runtime of the loop is

$$\begin{aligned}\sum_{t=1}^{T}\sum_{s=1}^{R_t} O(B_t \log B_t) &= O\left(\sum_{t=1}^{T} R_t \cdot (C_2 k/R_t^2)\log(C_2 k)\right)\\ &= O(k\log k)\cdot\sum_{t=1}^{T} 1/R_t\\ &= O(k\log k).\end{aligned}$$

The runtime for construction of the partition $S_1 \cup S_2 \cup \ldots \cup S_T$ in line 13 is $O((\sum_{t=1}^{T} R_t)|S|\log|S|) = O(R_T k \log k)$ by Lemma III.9 and the fact that $\sum_{t=1}^{T} R_t = O(R_T)$. We now note that since $T = \frac{1}{1-\delta}\log_2\log(k+1) + O(1)$, then

$$\begin{aligned}R_T = C_1 2^T &= C_1 2^{\frac{1}{1-\delta}\log_2\log k + O(1)}\\ &= O(\log_2^{1/(1-\delta)}(k+1))\\ &= O(\log_2^{1+2\delta}(k+1)),\end{aligned} \tag{15}$$

where we used the fact that $1/(1-\delta) \le 1 + 2\delta$ for $\delta \in (0, 1/2)$. Thus, the runtime for construction of the partition $S_1 \cup S_2 \cup \ldots \cup S_T$ in line 13 is $O(k\log^{2+2\delta} k)$.

By Lemma II.8 each invocation of ESTIMATEVALUES takes time $O((||\chi^{(r,t)}||_0 \log n + F B_t \log n)\cdot R_t) = O(R_t k \log n + R_t B_t \log n)$, as $F = O(1)$ by choice of parameters in line 25 of Algorithm 2. The total runtime per iteration in lines 16-22 is thus

$$\begin{aligned}&\sum_{t=1}^{T} O(R_t k \log n + R_t B_t \log n)\\ &= O\left(k R_T \log n + \sum_{t=1}^{T} B_t R_t \log n\right)\\ &= O(k R_T \log n) + O\left(\sum_{t=1}^{T} k/R_t\right)\log n,\end{aligned}$$

where the first transition follows since $\sum_{t=1}^{T} R_t = O(R_T)$, as R_t grow geometrically, and the second transition follows since $B_t = C_2 k/R_t^2$ by the setting of parameters. We now note that $\sum_{t=1}^{T} k/R_t = O(k)$ since R_t grow geometrically, and thus the expression on the last line above is $O(k R_T \log n) = k\log^{2+2\delta} n$ by (15). Since the loop in lines 16-22 proceeds over $O(\log n)$ iterations, the final runtime bound is $k\log^{3+2\delta} n$, as required (after rescaling δ).

Finally, lines 27-31 take $O(\frac{1}{\epsilon} k \log n)$ time for the invocation of HASHTOBINS by Lemma II.7 and $O(\frac{1}{\epsilon} k \log n)$ time for ESTIMATEVALUES by Lemma II.8. Putting the bounds above together, we obtain the runtime of $O(k\log^{3+2\delta} n) + O(\frac{1}{\epsilon} k \log n)$, as required. □

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.

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

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

[CKPS16] Xue Chen, Daniel M. Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 741–750, 2016.

[CKSZ17] Volkan Cevher, Michael Kapralov, Jonathan Scarlett, and Amir Zandieh. An adaptive sublinear-time block sparse Fourier transform, https://arxiv.org/abs/1702.01286. In *STOC*, 2017.

[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.

[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. In *51st Annual Allerton Conference on Communication, Control, and Computing, Allerton 2013, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, pages 1258–1265, 2013.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-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 fixed dimension. *FOCS*, 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.

[Kap16] Michael Kapralov. Sparse Fourier Transform in any constant dimension with nearly-optimal sample complexity in sublinear time (available as an arxiv report at http://arxiv.org/abs/1604.00845). *STOC*, 2016.

[Kap17] Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average, https://arxiv.org/abs/1708.04544. 2017.

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

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

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

[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.

[Pri11] Eric Price. Efficient sketches for the set query problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 41–56, 2011.

[PS15] Eric Price and Zhao 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.