Sample-Optimal Average-Case Sparse Fourier Transform in Two Dimensions

Badih Ghazi   Haitham Hassanieh   Piotr Indyk   Dina Katabi   Eric Price
Lixin Shi
Massachusetts Institute of Technology

{badih,haithamh,indyk,dk,ecprice,lixshi}@mit.edu

Abstract

We present the first sample-optimal sublinear time algorithms for the sparse Discrete Fourier Transform over a two-dimensional $\sqrt{n} \times \sqrt{n}$ grid. Our algorithms are analyzed for *average case* signals. For signals whose spectrum is exactly sparse, our algorithms use $O(k)$ samples and run in $O(k \log k)$ time, where k is the expected sparsity of the signal. For signals whose spectrum is approximately sparse, our algorithm uses $O(k \log n)$ samples and runs in $O(k \log^2 n)$ time; the latter algorithm works for $k = \Theta(\sqrt{n})$. The number of samples used by our algorithms matches the known lower bounds for the respective signal models.

By a known reduction, our algorithms give similar results for the one-dimensional sparse Discrete Fourier Transform when n is a power of a small composite number (e.g., $n = 6^t$).

1   Introduction

The Discrete Fourier Transform (DFT) is a powerful tool used in many domains. Multimedia data sets, including video and images, are typically processed in the frequency domain to compress the data [Wal91, HPN97, BK95]. Medicine and biology rely on the Fourier transform to analyze the output of a variety of tests and experiments including MRI [Nis10], NMR [MEH09] and ultrasound imaging [KS01]. Other applications include astronomy and radar systems.

The fastest known algorithm for computing the DFT is the Fast Fourier Transform (FFT). It computes the DFT of a signal with size n in $O(n \log n)$ time. Although it is not known whether this algorithm is optimal, any general algorithm for computing the exact DFT must take time at least proportional to its output size, i.e., $\Omega(n)$. In many applications, however, most of the Fourier coefficients of a signal are small or equal to zero, i.e., the output of the DFT is (approximately) *sparse*. This sparsity provides the rationale underlying compression schemes for image and video signals such as JPEG and MPEG. In fact, all of the aforementioned applications involve sparse data.

For sparse signals, the $\Omega(n)$ lower bound for the complexity of DFT no longer applies. If a signal has a small number k of nonzero Fourier coefficients—the *exactly k-sparse* case—the output of the Fourier transform can be represented succinctly using only k coefficients. Hence, for such signals, one may hope for a DFT algorithm whose runtime is sublinear in the signal size n. Even in the more general *approximately k-sparse* case, it is possible in principle to find the large components of its Fourier transform in sublinear time.

The past two decades have witnessed significant advances in sublinear sparse Fourier algorithms. The first such algorithm (for the Hadamard transform) appeared in [KM91] (building on [GL89]). Since then,

several sublinear sparse Fourier algorithms for complex inputs have been discovered [Man92, GGI$^+$02, AGS03, GMS05, Iwe10, Aka10, HIKP12b, HIKP12a, LWC12, BCG$^+$12, HAKI12]. The most efficient of those algorithms[1], given in [HIKP12a], offers the following performance guarantees:

- For signals that are exactly k-sparse, the algorithm runs in $O(k \log n)$ time.
- For the approximately sparse signals, the algorithm runs in $O(k \log n \log(n/k))$ time.

Although the aforementioned algorithms are very efficient, they nevertheless suffer from limitations. Perhaps the main limitation is that their sample complexity bounds are equal to the their running times. In particular, the sample complexity of the first algorithm (for the exactly k-sparse case) is $\Theta(k \log n)$, while the sample complexity of the second algorithm (approximately sparse) is $\Theta(k \log(n) \log(n/k))$. The first bound is suboptimal by a logarithmic factor, as it is known that one can recover any signal with k nonzero Fourier coefficients from $O(k)$ samples [AT08], albeit in super-linear time. The second bound is a logarithmic factor away from the lower bound of $\Omega(k \log(n/k))$ [PW11] established for non-adaptive algorithms[2]; a slightly weaker lower bound of $\Omega(k \log(n/k)/\log \log n)$ applies to adaptive algorithms as well [HIKP12a]. In most applications, low sample complexity is at least as important as efficient running time, as it implies reduced signal acquisition or communication cost.

Another limitation of the prior algorithms is that most of them are designed for one-dimensional signals. This is unfortunate, since multi-dimensional instances of DFT are often particularly sparse. This situation is somewhat alleviated by the fact that the two- dimensional DFT over $p \times q$ grids can be reduced to the one-dimensional DFT over a signal of length pq [GMS05, Iwe12]. However, the reduction applies only if p and q are relatively prime, which 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 [HIKP12b] algorithm (outlined in the appendix) has roughly $O(k \log^3 n)$ time and sample complexity.

Our results   In this paper, we present the first sample-optimal sublinear time algorithms for the Discrete Fourier Transform over a two- dimensional $\sqrt{n} \times \sqrt{n}$ grid. Unlike the aforementioned results, our algorithms are analyzed in the *average case*. Our input distributions are natural. For the exactly sparse case, we assume the Bernoulli model: each spectrum coordinate is nonzero with probability k/n, in which case the entry assumes an arbitrary value predetermined for that position[3]. For the approximately sparse case, we assume that the spectrum $\widehat{x}$ of the signal is a sum of two vectors: the signal vector, chosen from the Bernoulli distribution, and the noise vector, chosen from the Gaussian distribution (see Section §2 Preliminaries for the complete definition). These or similar[4] distributions are often used as test cases for empirical evaluations of sparse Fourier Transform algorithms [IGS07, HIKP12b, LWC12] or theoretical analysis of their performance [LWC12].

The algorithms succeed with a constant probability. The notion of success depends on the scenario considered. For the exactly sparse case, an algorithm is successful if it recovers the spectrum exactly. For

---

[1]See the discussion in the Related Work section.

[2]An algorithm is *adaptive* if it selects the samples based on the values of the previously sampled coordinates. If the positions of the samples are chosen in advance of the sampling process, the algorithm is called *non-adaptive*. All algorithms given in this paper are non-adaptive.

[3]Note that this model subsumes the scenario where the values of the nonzero coordinates are chosen i.i.d. from some distribution.

[4]A popular alternative is to use the hypergeometric distribution over the set of nonzero entries instead of the Bernoulli distribution. The advantage of the former is that it yields vectors of sparsity *exactly* equal to k. In this paper we opted for the Bernoulli model since it is simpler to analyze. However, both models are quite similar. In particular, for large enough k, the actual sparsity of vectors in the Bernoulli model is sharply concentrated around k.

the approximately sparse case, the algorithm is successful if it reports a signal with spectrum $\widehat{z}$ such that

$$\|\widehat{z} - \widehat{x}\|_2^2 = O(\sigma^2 n) + \|\widehat{x}\|_2^2 / n^c \tag{1}$$

where σ^2 denotes the variance of the normal distributions defining each coordinate of the noise vector, and where c is any constant. Note that any k-sparse approximation to $\widehat{x}$ has error $\Omega(\sigma^2 n)$ with overwhelming probability, and that the second term in the bound in Equation 1 is subsumed by the first term as long as the signal-to-noise ratio is at most polynomial, i.e., $\|\widehat{x}\|_2 \leq n^{O(1)}\sigma$. See Section §2 for further discussion.

The running time and sample complexity bounds are depicted in the following table. We assume that $\sqrt{n}$ is a power of 2.

Input	Samples	Time	Assumptions
Sparse	k	$k \log k$	$k = O(\sqrt{n})$
Sparse	k	$k \log k$ $+k(\log \log n)^{O(1)}$	
Approx. sparse	$k \log n$	$k \log^2 n$	$k = \Theta(\sqrt{n})$

The key feature of our algorithms is that their sample complexity bounds are optimal, at least in the non-adaptive case. For the exactly sparse case, the lower bound of $\Omega(k)$ is immediate. For the approximately sparse case, we note that the $\Omega(k \log(n/k))$ lower bound of [PW11] holds even if the spectrum is the sum of a k-sparse signal vector in $\{0, 1, -1\}^n$ and Gaussian noise. The latter is essentially a special case of the distributions handled by our algorithm, and we give a full reduction in Appendix A. From the running time perspective, our algorithms are slightly faster than those in [HIKP12a], with the improvement occurring for low values of k.

An additional feature of the first algorithm is its simplicity and therefore its low "big-Oh" overhead. Our preliminary experiments on random sparse data indicate that the algorithm for exactly sparse case yields substantial improvement over 2D FFTW, a highly efficient implementation of 2D FFT. In particular, for $n = 2^{22}$ (a 2048×2048 signal) and $k = 1024$, the algorithm is $100\times$ faster than 2D FFTW. To the best of our knowledge, this is the first implementation of a 2D sparse FFT algorithm. For the same n and k, the algorithm has a comparable running time ($1.5\times$ faster) to the 1D exactly sparse FFT in [HIKP12a] while using $8\times$ fewer samples. We expect that the algorithm or its variant will be efficient on non-random data as well, since the algorithm can randomize the positions of the coefficients using random two-dimensional affine transformations (cf. Appendix B). Even though the resulting distribution is not fully random, it has been observed that random affine transformations work surprisingly well on real data [MV08].

Our techniques Our first algorithm for k-sparse signals is based on the following idea. Recall that one way to compute the two-dimensional DFT of a signal x is to apply the one-dimensional DFT to each column and then to each row. Suppose that $k = a\sqrt{n}$ for $a < 1$. In this case, the expected number of nonzero entries in each row is less than 1. If *every* row contained exactly one nonzero entry, then the DFT could be computed via the following two step process. In the first step, we select the first two columns of x, denoted by $u^{(0)}$ and $u^{(1)}$, and compute their DFTs $\widehat{u}^{(0)}$ and $\widehat{u}^{(1)}$. Let j_i be the index of the unique nonzero entry in the i-th row of $\widehat{x}$, and let a be its value. Observe that $\widehat{u}_i^{(0)} = a$ and $\widehat{u}_i^{(1)} = a\omega^{-j_i}$ (where ω is a primitive $\sqrt{n}$-th root of unity), as these are the first two entries of the inverse Fourier transform of a 1-sparse signal ae_{j_i}. Thus, in the second step, we can retrieve the value of the nonzero entry, equal to $\widehat{u}_i^{(0)}$, as well as the index j_i from the phase of the ratio $\widehat{u}_i^{(1)}/\widehat{u}_i^{(0)}$ (this technique was introduced in [HIKP12a, LWC12] and was referred to as the "OFDM trick"). The total time is dominated by the cost of the two DFTs of the columns, which
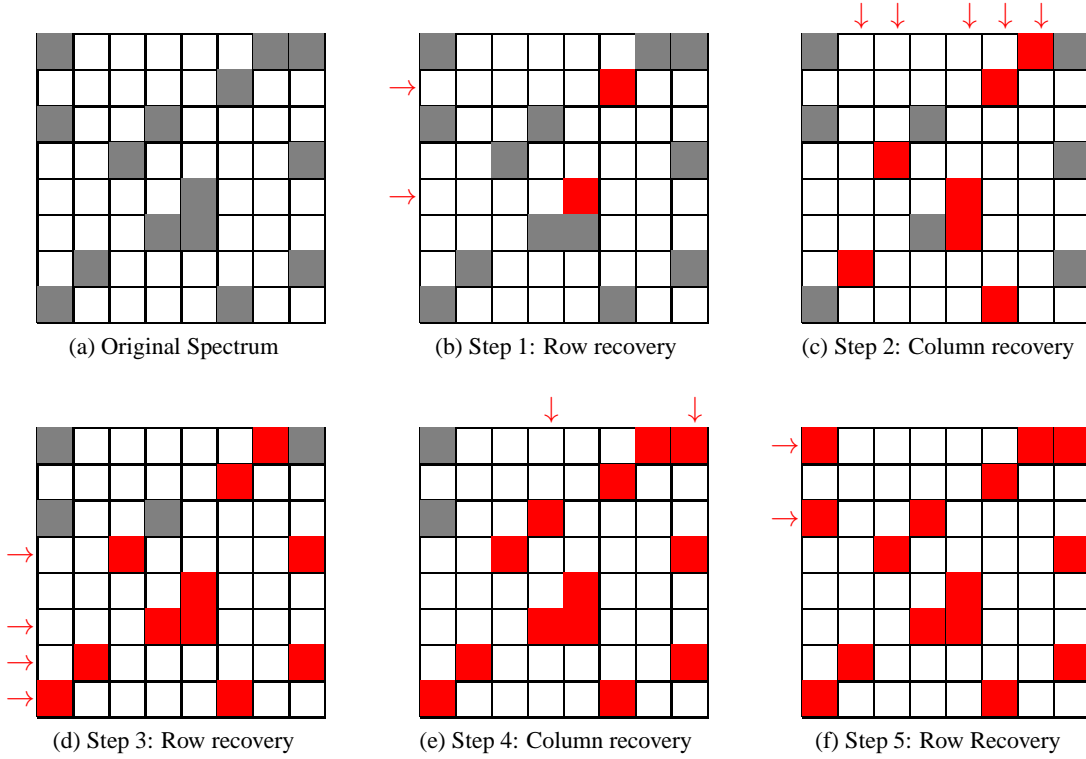
(a) Original Spectrum (b) Step 1: Row recovery (c) Step 2: Column recovery

(d) Step 3: Row recovery (e) Step 4: Column recovery (f) Step 5: Row Recovery

Figure 1: An illustration of the "peeling" recovery process on an 8×8 signal with 15 nonzero frequencies. In each step, the algorithm recovers all 1-sparse columns and rows (the recovered entries are depicted in red). The process converges after a few steps.

is $O(\sqrt{n} \log n)$. Since the algorithm queries only a constant number of columns, its sample complexity is $O(\sqrt{n})$.

In general, the distribution of the nonzero entries over the rows can be non-uniform. Thus, our actual algorithm alternates the above recovery process between the columns and rows (see Figure 1 for an illustration). Since the OFDM trick works only on 1-sparse columns/rows, we check the 1-sparsity of each column/row by sampling a constant number of additional entries. We then show that, as long as the sparsity constant a is small enough, this process recovers all entries in a logarithmic number steps with constant probability. The proof uses the fact that the probability of the existence of an "obstructing configuration" of nonzero entries which makes the process deadlocked (e.g., see Figure 2) is upper bounded by a small constant.

The algorithm is extended to the case of $k = o(\sqrt{n})$ via a reduction. Specifically, we subsample the signal x by the reduction ratio $R = \alpha\sqrt{n}/k$ for some small enough constant α in each dimension. The subsampled signal x' has dimension $\sqrt{m} \times \sqrt{m}$, where $\sqrt{m} = \frac{k}{\alpha}$. Since subsampling in time domain corresponds to "spectrum folding", i.e., adding together all frequencies with indices that are equal modulo $\sqrt{m}$, the nonzero entries of $\widehat{x}$ are mapped into the entries of $\widehat{x'}$. It can be seen that, with constant probability, the mapping is one-to-one. If this is the case, we can use the earlier algorithm for sparse DFT to compute the nonzero frequencies in $O(\sqrt{m} \log m) = O(\sqrt{k} \log k)$ time, using $O(k)$ samples. We then use the OFDM trick to identify the positions of those frequencies in $\widehat{x}$.
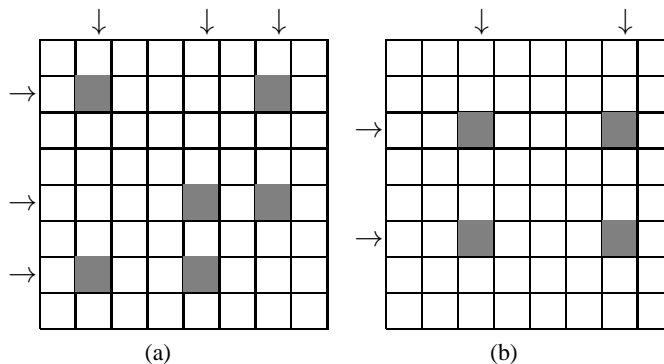
4

Figure 2: Examples of obstructing sequences of nonzero entries. None of the remaining rows or columns has a sparsity of 1.

Our second algorithm for the exactly sparse case works for all values of k. The main idea behind it is to decode rows/columns with higher sparsity than 1. First, we give a *deterministic*, *worst-case* algorithm for 1-dimensional sparse Fourier transforms that takes $O(k^2 + k(\log\log n)^{O(1)})$ time. This algorithm uses the relationship between sparse recovery and syndrome decoding of Reed-Solomon codes (due to [AT08]). Although a simple application of the decoder yields $O(n^2)$ decoding time, we show that by using appropriate numerical subroutines one can in fact recover a k-sparse vector from $O(k)$ samples in time $O(k^2 + k(\log\log n)^{O(1)})^5$. In particular, we use Berlekamp-Massey's algorithm for constructing the error-locator polynomial and Pan's algorithm for finding its roots. For our fast average-case, 2-dimensional sparse Fourier transform algorithm, we fold the spectrum into $B = \frac{k}{C\log k}$ bins for some large constant C. Since the positions of the k nonzero frequencies are random, it follows that each bin receives $t = \Theta(\log k)$ frequencies with high probability. We then take $\Theta(t)$ samples of the time domain signal corresponding to each bin, and recover the frequencies corresponding to those bins in $O(t^2 + t(\log\log n)^{O(1)})$ time per bin, for a total time of $O(k\log k + k(\log\log n)^{O(1)})$.

The above approach works as long as the number of nonzero coefficients per column/row are highly concentrated. However, this is not the case for $k \ll \sqrt{n}\log n$. We overcome this difficulty by replacing a row by a sequence of rows. A technical difficulty is that the process might lead to collisions of coefficients. We resolve this issue by using a two level procedure, where the first level returns the syndromes of colliding coefficients as opposed to the coefficients themselves; the syndromes are then decoded at the second level.

Our third algorithm works for *approximately* sparse data, at sparsity $\Theta(\sqrt{n})$. Its general outline mimics that of the first algorithm. Specifically, it alternates between decoding columns and rows, assuming that they are 1-sparse. The decoding subroutine itself is similar to that of [HIKP12a] and uses $O(\log n)$ samples. The subroutine first checks whether the decoded entry is large; if not, the spectrum is unlikely to contain any large entry, and the subroutine terminates. The algorithm then subtracts the decoded entry from the column and checks whether the resulting signal contains no large entries in the spectrum (which would be the case if the original spectrum was approximately 1-sparse and the decoding was successful). The check is done by sampling $O(\log n)$ coordinates and checking whether their sum of squares is small. To prove that this check works with high probability, we use the fact that a collection of random rows of the Fourier matrix is likely to satisfy the Restricted Isometry Property (RIP) of [CT06].

A technical difficulty in the analysis of the algorithm is that the noise accumulates in successive itera-

---

[5]We note that, for $k = o(\log n)$, this is the fastest known *worst-case* algorithm for the exactly sparse DFT.

tions. This means that a $1/\log^{O(1)} n$ fraction of the steps of the algorithm will fail. However, we show that the dependencies are "local", which means that our analysis still applies to a vast majority of the recovered entries. We continue the iterative decoding for $\log \log n$ steps, which ensures that all but a $1/\log^{O(1)} n$ fraction of the large frequencies are correctly recovered. To recover the remaining frequencies, we resort to algorithms with worst-case guarantees.

Extensions Our algorithms have natural extensions to dimensions higher than 2. We do not include them in this paper as the description and analysis are rather cumbersome.

While no optimal result is known for the 1-dimensional case, one can achieve optimal sample complexity and efficient robust recovery in the $\log n$-dimensional (Hadamard) case ([Lev93], see also Appendix C.2 of [Gol99]). Our result demonstrates that even two dimensions give enough flexibility for optimal sample complexity in the average case. Due to the equivalence between the two-dimensional case and the one-dimensional case where n is a product of different prime powers [GMS05, Iwe12], our algorithm also gives optimal sample complexity bounds for e.g., $n = 6^t$ in the average case.

1.1   Related work

As described in the introduction, currently the most efficient algorithms for computing the sparse DFT are due to [HIKP12a]. For signals that are exactly k-sparse, the first algorithm runs in $O(k \log n)$ time. For approximately sparse signals, the second algorithm runs in $O(k \log n \log(n/k))$ time. Formally, the latter algorithm works for any signal x, and computes an approximation vector $\widehat{x}'$ that satisfies the ℓ_2/ℓ_2 approximation guarantee, i.e., $\|\widehat{x} - \widehat{x}'\|_2 \leq C \min_{k\text{-sparse } y} \|\widehat{x} - y\|_2$, where C is some approximation factor and the minimization is over k-sparse signals. Note that this guarantee generalizes that of Equation (1).

We also mention another efficient algorithm, due to [LWC12], designed for the exactly k-sparse model. The average case analysis presented in that paper shows that the algorithm has $O(k)$ expected sample complexity and runs in $O(k \log k)$ time. However, the algorithm assumes that the input signal x is specified as a *function* over an interval $[0, 1]$ that can be sampled at arbitrary positions, as opposed to a given discrete sequence of n samples as in our case. Thus, although very efficient, that algorithm does not solve the Discrete Fourier Transform problem.

2   Preliminaries

This section introduces the notation, assumptions and definitions used in the rest of this paper.

Notation Throughout the paper we assume that $\sqrt{n}$ is a power of 2. We use $[m]$ to denote the set $\{0, \ldots, m-1\}$, and $[m] \times [m] = [m]^2$ to denote the $m \times m$ grid $\{(i,j) : i \in [m], j \in [m]\}$. We define $\omega = e^{-2\pi \mathbf{i}/\sqrt{n}}$ to be a primitive $\sqrt{n}$-th root of unity and $\omega' = e^{-2\pi \mathbf{i}/n}$ to be a primitive n-th root of unity. For any complex number a, we use $\phi(a) \in [0, 2\pi)$ to denote the *phase* of a.   For a 2D matrix $x \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, its support is denoted by $\mathrm{supp}(x) \subseteq [\sqrt{n}] \times [\sqrt{n}]$. We use $\|x\|_0$ to denote $|\mathrm{supp}(x)|$, the number of nonzero coordinates of x. Its 2D Fourier spectrum is denoted by $\widehat{x}$, with

$$\widehat{x}_{i,j} = \frac{1}{\sqrt{n}} \sum_{l \in [\sqrt{n}]} \sum_{m \in [\sqrt{n}]} \omega^{il+jm} x_{l,m}.$$

Similarly, if y is a frequency-domain signal, its inverse Fourier transform is denoted by $\check{y}$.

Definitions The paper uses the comb filter used in [Iwe10, HIKP12b] (cf. [Man92]). The filter can be generalized to 2 dimensions as follows:

Given $(\tau_r, \tau_c) \in [\sqrt{n}] \times [\sqrt{n}]$, and B_r, B_c that divide $\sqrt{n}$, then for all $(i, j) \in [B_r] \times [B_c]$ set

$$y_{i,j} = x_{i(\sqrt{n}/B_r)+\tau_r, j(\sqrt{n}/B_c)+\tau_c}.$$

Then, compute the 2D DFT $\hat{y}$ of y. Observe that $\hat{y}$ is a folded version of $\hat{x}$:

$$\hat{y}_{i,j} = \sum_{l \in [\sqrt{n}/B_r]} \sum_{m \in [\sqrt{n}/B_c]} \hat{x}_{lB_r+i, mB_c+j} \omega^{-\tau_r(i+lB_r)-\tau_c(j+mB_c)}.$$

Distributions In the exactly sparse case, we assume a Bernoulli model for the support of $\hat{x}$. This means that for all $(i, j) \in [\sqrt{n}] \times [\sqrt{n}]$,
$$\Pr\{(i, j) \in \text{supp}(\hat{x})\} = k/n$$

and thus $\mathbb{E}[|\text{supp}(\hat{x})|] = k$. We assume an unknown predefined matrix $a_{i,j}$ of values in $\mathbb{C}$; if $\hat{x}_{i,j}$ is selected to be nonzero, its value is set to $a_{i,j}$.

In the approximately sparse case, we assume that the signal $\hat{x}$ is equal to $\widehat{x^*} + \widehat{w} \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, where $\widehat{x^*}_{i,j}$ is the "signal" and $\widehat{w}$ is the "noise". In particular, $\widehat{x^*}$ is drawn from the Bernoulli model, where $\widehat{x^*}_{i,j}$ is drawn from $\{0, a_{i,j}\}$ at random independently for each (i, j) for some values $a_{i,j}$ and with $\mathbb{E}[|\text{supp}(\widehat{x^*})|] = k$. We also require that $|a_{i,j}| \geq L$ for some parameter L. $\widehat{w}$ is a complex Gaussian vector with variance σ^2 in both the real and imaginary axes independently on each coordinate; we notate this as $\widehat{w} \sim N_{\mathbb{C}}(0, \sigma^2 I_n)$. We will need that $L = C\sigma\sqrt{n/k}$ for a sufficiently large constant C, so that $\mathbb{E}[\|\widehat{x^*}\|_2^2] \geq C\,\mathbb{E}[\|\widehat{w}\|_2^2]$.

3   Basic Algorithm for the Exactly Sparse Case

The algorithm for the noiseless case depends on the sparsity k where $k = \mathbb{E}[|\text{supp}(\hat{x})|]$ for a Bernoulli distribution of the support.

3.1   Basic Exact Algorithm: $k = \Theta(\sqrt{n})$

In this section, we focus on the regime $k = \Theta(\sqrt{n})$. Specifically, we will assume that $k = a\sqrt{n}$ for a (sufficiently small) constant $a > 0$.

The algorithm BASICEXACT2DSFFT is described as Algorithm 3.1. The key idea is to fold the spectrum into bins using the comb filter defined in §2 and estimate frequencies which are isolated in a bin. The algorithm takes the FFT of a row and as a result frequencies in the same columns will get folded into the same row bin. It also takes the FFT of a column and consequently frequencies in the same rows wil get folded into the same column bin. The algorithm then uses the OFDM trick introduced in [HIKP12a] to recover the columns and rows whose sparsity is 1. It iterates between the column bins and row bins, subtracting the recovered frequencies and estimating the remaining columns and rows whose sparsity is 1. An illustration of the algorithm running on an 8×8 signal with 15 nonzero frequencies is shown in Fig. 1 in Section 1. The algorithm also takes a constant number of extra FFTs of columns and rows to check for collisions within a bin and avoid errors resulting from estimating bins where the sparsity is greater than 1. The algorithm uses three functions:

- FOLDTOBINS. This procedure folds the spectrum into $B_r \times B_c$ bins using the comb filter described §2.

- BASICESTFREQ. Given the FFT of rows or columns, it estimates the frequency in the large bins. If there is no collision, i.e. if there is a single nonzero frequency in the bin, it adds this frequency to the result $\widehat{w}$ and subtracts its contribution to the row and column bins.

- BASICEXACT2DSFFT. This performs the FFT of the rows and columns and then iterates BASICEST-FREQ between the rows and columns until is recovers $\widehat{x}$.

Analysis of BASICEXACT2DSFFT

Lemma 3.1. *For any constant $\alpha > 0$, if $a > 0$ is a sufficiently small constant, then assuming that all 1-sparsity tests in the procedure BASICESTFREQ are correct, the algorithm reports the correct output with probability at least $1 - O(\alpha)$.*

Proof. The algorithm fails if there is a pair of nonzero entries in a column or row of $\widehat{x}$ that "survives" $t_{max} = C \log n$ iterations. For this to happen there must be an "obstructing" sequence of nonzero entries $p_1, q_1, p_2, q_2 \ldots p_t$, $3 \leq t \leq t_{max}$, such that for each $i \geq 1$, p_i and q_i are in the same column ("vertical collision"), while q_i and p_{i+1} are in the same row ("horizontal collision"). Moreover, it must be the case that either the sequence "loops around", i.e., $p_1 = p_t$, or $t > t_{max}$. We need to prove that the probability of either case is less than α. We focus on the first case; the second one is similar.

Assume that there is a sequence $p_1, q_1, \ldots p_{t-1}, q_{t-1}$ such that the elements in this sequence are all distinct, while $p_1 = p_t$. If such a sequence exists, we say that the event E_t holds. The number of sequences satisfying E_t is at most $\sqrt{n}^{2(t-1)}$, while the probability that the entries corresponding to the points in a specific sequence are nonzero is at most $(k/n)^{2(t-1)} = (a/\sqrt{n})^{2(t-1)}$. Thus the probability of E_t is at most

$$\sqrt{n}^{2(t-1)} \cdot (a/\sqrt{n})^{2(t-1)} = a^{2(t-1)}.$$

Therefore, the probability that one of the events $E_1, \ldots, E_{t_{max}}$ holds is at most $\sum_{t=3}^{\infty} a^{2(t-1)} = a^4/(1-a^2)$, which is smaller than α for a small enough. $\square$

Lemma 3.2. *The probability that any 1-sparsity test invoked by the algorithm is incorrect is at most $O(1/n^{(c-5)/2})$.*

To prove Lemma 3.2, we first need the following lemma.

Lemma 3.3. *Let $y \in \mathbb{C}^m$ be drawn from a permutation invariant distribution with $r \geq 2$ nonzero values. Let $T = [2c]$. Then the probability that there exists a y' such that $\|y'\|_0 \leq 1$ and $(\widehat{y} - \widehat{y}')_T = 0$ is at most $c\left(\frac{c}{m-r}\right)^{c-2}$.*

Proof. Let $A = F_T$ be the first $2c$ rows of the inverse Fourier matrix. Because any $2c \times 2c$ submatrix of A is Vandermonde and hence non-singular, the system of linear equations

$$Az = b$$

has at most one c-sparse solution in z, for any b.

If $r \leq c - 1$, then $\|y - y'\|_0 \leq c$ so $A(y - y') = 0$ implies $y - y' = 0$. But $r \geq 2$ so $\|y - y'\|_0 > 0$. This is a contradiction, so if $r < c$ then the probability that $(\widehat{y} - \widehat{y}')_T = 0$ is zero. Henceforth, we assume $r \geq c$.

8

procedure FOLDTOBINS(x, B_r, B_c, τ_r, τ_c)

    $y_{i,j} = x_{i(\sqrt{n}/B_r)+\tau_r, j(\sqrt{n}/B_c)+\tau_c}$ for $(i,j) \in [B_r] \times [B_c]$,
    return $\widehat{y}$, the DFT of y
end procedure
procedure BASICESTFREQ($\widehat{u}^{(T)}$, $\widehat{v}^{(T)}$, T, IsCol)

    $\widehat{w} \leftarrow 0$.
    Compute $J = \{j : \sum_{\tau \in T} |\widehat{u}_j^{(\tau)}| > 0\}$.
    for $j \in J$ **do**
        $b \leftarrow \widehat{u}_j^{(1)}/\widehat{u}_j^{(0)}$.
        $i \leftarrow \text{round}(\phi(b)\frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$.   ▷ $\phi(b)$ is the phase of b.
        $s \leftarrow \widehat{u}_j^{(0)}$.

                                           ▷ Test whether the row or column is 1-sparse

        if $\left(\sum_{\tau \in T} |\widehat{u}_j^{(\tau)} - s\omega^{-\tau i}| == 0\right)$ **then**
            if IsCol **then**   ▷ whether decoding column or row
                $\widehat{w}_{i,j} \leftarrow s$.
            else
                $\widehat{w}_{j,i} \leftarrow s$.
            end if
            for $\tau \in T$ **do**
                $\widehat{u}_j^{(\tau)} \leftarrow 0$
                $\widehat{v}_i^{(\tau)} \leftarrow \widehat{v}_i^{(\tau)} - s\omega^{-\tau i}$
            end for
        end if
    end for
    return $\widehat{w}$, $\widehat{u}^{(T)}$, $\widehat{v}^{(T)}$
end procedure
procedure BASICEXACT2DSFFT(x, k)

    $T \leftarrow [2c]$ ▷ We set $c \geq 6$
    for $\tau \in T$ **do**
        $\widehat{u}^{(\tau)} \leftarrow$ FOLDTOBINS(x, $\sqrt{n}$, 1, 0, τ).
        $\widehat{v}^{(\tau)} \leftarrow$ FOLDTOBINS(x, 1, $\sqrt{n}$, τ, 0).
    end for
    $\widehat{z} \leftarrow 0$
    for $t \in [C \log n]$ **do**   ▷ $\widehat{u}^{(T)} := \{\widehat{u}^{(\tau)} : \tau \in T\}$
        $\{\widehat{w}, \widehat{u}^{(T)}, \widehat{v}^{(T)}\} \leftarrow$ BASICESTFREQ($\widehat{u}^{(T)}$, $\widehat{v}^{(T)}$, T, true).
        $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
        $\{\widehat{w}, \widehat{v}^{(T)}, \widehat{u}^{(T)}\} \leftarrow$ BASICESTFREQ($\widehat{v}^{(T)}$, $\widehat{u}^{(T)}$, T, false).
        $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
    end for
    return $\widehat{z}$
end procedure

Algorithm 3.1: Basic Exact 2D sparse FFT algorithm for $k = \Theta(\sqrt{n})$

When drawing y, first place $r - (c - 1)$ coordinates into u then place the other $c - 1$ values into v, so that $y = u + v$. Condition on u, so v is a permutation distribution over $m - r + c - 1$ coordinates. We know there exists at most one c-sparse vector w with $Aw = -Au$. Then

$$
\begin{aligned}
&\Pr_y[\exists y' : A(y - y') = 0 \text{ and } \|y'\|_0 \leq 1] \\
&= \Pr_v[\exists y' : A(v - y') = -Au \text{ and } \|y'\|_0 \leq 1] \\
&\leq \Pr_v[\exists y' : v - y' = w \text{ and } \|y'\|_0 \leq 1] = \Pr_v[\|v - w\|_0 \leq 1] \\
&\leq \Pr_v[|\operatorname{supp}(v) \triangle \operatorname{supp}(w)| \leq 1] \\
&< \frac{m - r + c - 1}{\binom{m-r+c-1}{c-1}} < c \left(\frac{c}{m - r} \right)^{c-2}
\end{aligned}
$$

where the penultimate inequality follows from considering the cases $\|w\|_0 \in \{c-2, c-1, c\}$ separately. $\qquad\square$

We now proceed with the proof of Lemma 3.2 .

Proof. W.L.O.G. consider the row case. Let y be the jth row of $\widehat{x}$. Note that $\widehat{u}_j^{(\tau)} = \widehat{y}_\tau$. Observe that with probability at least $1 - 1/n^c$ we have $\|y\|_0 \leq r$ for $r = c \log n$. Moreover, the distribution of y is permutation-invariant, and the test in BASICESTFREQ corresponds to checking whether $(\widehat{y} - \widehat{y}')_T = 0$ for some 1-sparse $y' = ae_i$. Hence, Lemma 3.3 with $m = \sqrt{n}$ implies the probability that any specific test fails is less than $c(2c/\sqrt{n})^{c-2}$. Taking a union bound over the $\sqrt{n} \log n$ total tests gives a failure probability of $4c^3 \log n (2c/\sqrt{n})^{c-4} < O(1/n^{(c-5)/2})$. $\qquad\square$

Theorem 3.4. *For any constant α, the algorithm* BASICEXACT2DSFFT *uses $O(\sqrt{n})$ samples, runs in time $O(\sqrt{n} \log n)$ and returns the correct vector $\widehat{x}$ with probablility at least $1 - O(\alpha)$ as long as a is a small enough constant.*

Proof. From Lemma 3.1 and Lemma 3.2, the algorithm returns the correct vector $\widehat{x}$ with probability at least $1 - O(\alpha) - O(n^{-(c-5)/2}) = 1 - O(\alpha)$ for $c > 5$.

The algorithm uses only $O(T) = O(1)$ rows and columns of x, which yields $O(\sqrt{n})$ samples. The running time is bounded by the time needed to perform $O(1)$ FFTs of rows and columns (in FOLDTOBINS) procedure, and $O(\log n)$ invocations of BASICESTFREQ. Both components take time $O(\sqrt{n} \log n)$. $\qquad\square$

3.2 Reduction to Basic Exact Algorithm: $k = o(\sqrt{n})$

Algorithm REDUCEEXACT2DSFFT, which is for the case where $k = o(\sqrt{n})$, is described in Algorithm 3.2). The key idea is to reduce the problem from the case where $k = o(\sqrt{n})$ to the case where $k = \Theta(\sqrt{n})$. To do that, we subsample the input time domain signal x by the reduction ratio $R = a\sqrt{n}/k$ for some small enough a. The subsampled signal x' has dimension $\sqrt{m} \times \sqrt{m}$, where $\sqrt{m} = \frac{k}{a}$. This implies that the probability that any coefficient in $\widehat{x}'$ is nonzero is at most $R^2 \times k/n = a^2/k = (a^2/k) \times (k^2/a^2)/m = k/m$, since $m = k^2/a^2$. This means that we can use the algorithm BASICNOISE-LESS2DSFFT in subsection §3.1 to recover $\widehat{x}'$. Each of the entries of $\widehat{x}'$ is a frequency in $\widehat{x}$ which was folded into $\widehat{x}'$. We employ the same phase technique used in [HIKP12a] and subsection §3.1 to recover their original frequency position in $\widehat{x}$.

The algorithm uses 2 functions:

- REDUCETOBASICSFFT: This folds the spectrum into $O(k) \times O(k)$ dimensions and performs the reduction to BASICEXACT2DSFFT. Note that only the $O(k)$ elements of x' which will be used in BASICEXACT2DSFFT need to be computed.

- REDUCEEXACT2DSFFT: This invokes the reduction as well as the phase technique to recover $\widehat{x}$.

---

procedure REDUCETOBASICSFFT(x, R, τ_r, τ_c)
    Define $x'_{ij} = x_{iR+\tau_r, jR+\tau_c}$ ▷ With lazy evaluation
    return BASICEXACT2DSFFT(x', k)
end procedure
procedure REDUCEEXACT2DSFFT(x, k)
    $R \leftarrow \frac{a\sqrt{n}}{k}$, for some constant $a < 1$ such that $R | \sqrt{n}$.
    $\widehat{u}^{(0,0)} \leftarrow$ REDUCETOBASICSFFT($x, R, 0, 0$)
    $\widehat{u}^{(1,0)} \leftarrow$ REDUCETOBASICSFFT($x, R, 1, 0$)
    $\widehat{u}^{(0,1)} \leftarrow$ REDUCETOBASICSFFT($x, R, 0, 1$)
    $\widehat{z} \leftarrow 0$
    $L \leftarrow \text{supp}(\widehat{u}^{(0,0)}) \cap \text{supp}(\widehat{u}^{(1,0)}) \cap \text{supp}(\widehat{u}^{(0,1)})$
    for $(\ell, m) \in L$ **do**
        $b_r \leftarrow \widehat{u}^{(1,0)}_{\ell,m} / \widehat{u}^{(0,0)}_{\ell,m}$
        $i \leftarrow \text{round}(\phi(b_r)\frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$
        $b_c \leftarrow \widehat{u}^{(0,1)}_{\ell,m} / \widehat{u}^{(0,0)}_{\ell,m}$
        $j \leftarrow \text{round}(\phi(b_c)\frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$
        $\widehat{z}_{ij} \leftarrow \widehat{u}^{(0,0)}_{\ell,m}$
    end for
    return $\widehat{z}$
end procedure

Algorithm 3.2: Exact 2D sparse FFT algorithm for $k = o(\sqrt{n})$

Analysis of REDUCEEXACT2DSFFT

Lemma 3.5. *For any constant α, for sufficiently small a there is a one-to-one mapping of frequency coefficients from $\widehat{x}$ to $\widehat{x}'$ with probability at least $1 - \alpha$.*

Proof. The probability that there are at least 2 nonzero coefficients among the R^2 coefficients in $\widehat{x}$ that are folded together in $\widehat{x}'$, is at most

$$\binom{R^2}{2}(k/n)^2 < (a^2 n/k^2)^2 (k/n)^2 = a^4/k^2$$

The probability that this event holds for any of the m positions in $\widehat{x}'$ is at most $ma^4/k^2 = (k^2/a^2)a^4/k^2 = a^2$ which is less than α for small enough a. Thus, with probability at least $1 - \alpha$ any nonzero coefficient in $\widehat{x}'$ comes from only one nonzero coefficient in $\widehat{x}$. $\qquad\square$

Theorem 3.6. *For any constant $\alpha > 0$, there exists a constant $c > 0$ such that if $k < c\sqrt{n}$ then the algorithm* REDUCEEXACT2DSFFT *uses $O(k)$ samples, runs in time $O(k \log k)$ and returns the correct vector $\widehat{x}$ with probablility at least $1 - \alpha$.*

Proof. By Theorem 3.4 and the fact that each coefficient in $\widehat{x}'$ is nonzero with probability $O(1/k)$, each invocation of the function REDUCETOBASICSFFT fails with probability at most α. By Lemma 3.5, with probability at least $1 - \alpha$, we could recover $\widehat{x}$ correctly if each of the calls to REDTOBASICSFFT returns the correct result. By the union bound, the algorithm REDUCEEXACT2DSFFT fails with probability at most $\alpha + 3 \times \alpha = O(\alpha)$.

The algorithm uses $O(1)$ invocations of BASICEXACT2DSFFT on a signal of size $O(k) \times O(k)$ in addition to $O(k)$ time to recover the support using the OFDM trick. Noting that calculating the intersection L of supports takes $O(k)$ time, the stated number of samples and running time then follow directly from Theorem 3.4. $\qquad\square$

4 Algorithm for Exactly Sparse Case of any sparsity $k = O(n)$

4.1 Exact 1D Algorithm for $k = O(n)$

We will first present a deterministic algorithm for the one-dimensional exactly sparse case. The algorithm 1DSFFT, described in Alg. 4.1, computes the spectrum of a t-sparse signal $x \in \mathbb{C}^n$ and has worst case running time of $O(t^2 + t(\log \log n)^{O(1)})$ for $t = O(\log n)$. This deterministic algorithm has the fastest known worst case running time for $k = o(\log n)$. We will later use it to construct the algorithm EXACT1DSFFT that has the fastest known average case running time for $k = O(n)$.

---

procedure 1DSFFT(x, t)
    $\widehat{x} \leftarrow 0$
    $\{(f_i, v_i)\}_{i \in [l]} \leftarrow$ SIGNALFROMSYNDROME$(n, \{x_0, \cdots, x_{2t-1}\})$.
    $\widehat{x}_{f_i} \leftarrow v_i$ for all $i \in [l]$. $\triangleright\ l \leq t$ is result size
    return $\widehat{x}$
end procedure

procedure SIGNALFROMSYNDROME$(n, \{s_0, \cdots, s_{2t-1}\})$
    $\Lambda(z) \leftarrow$ BERLEKAMPMASSEY$(\{s_0, \cdots, s_{2t-1}\})$.
    $f \leftarrow$ PAN$(\Lambda(z))$. $\triangleright$ This finds $\{f : |\widehat{x}_f| > 0\}$ of length $l \leq t$
    $V \leftarrow$ VANDERMONDE$((\omega')^{f_0}, \cdots, (\omega')^{f_{l-1}})$
    $V^{-1} \leftarrow$ INVERSEVANDERMONDE(V)
    $v \leftarrow V^{-1}s_{[l]}$
    return $\{(f_i, v_i)\}_{i \in [l]}$
end procedure

Algorithm 4.1: Algorithm for computing the exact 1D FFT of a t-sparse signal of size n

1DSFFT is a wrapper for the "Signal From Syndrome" procedure SIGNALFROMSYNDROME which uses the following procedures:

- BERLEKAMPMASSEY: This function finds the coefficients of the error locator polynomial $\Lambda(z)$ based on the input *syndromes* $s_i = \sum_i \widehat{x}_i(\omega')^i$. The error locator polynomial, which only depends on the locations

of the nonzero frequency components of x, is given by:

$$\Lambda(z) = \prod_{\ell=1}^{t} \left(1 - z(\omega')^{f_\ell} \right) \tag{2}$$

Constructing the error locator polynomial is a commonly used step in decoding Reed-Solomon codes [MS77] In our case, the main difference is that the coefficients of $\Lambda(z)$ lie in $\mathbb{C}$ whereas they lie in a finite field in the case of Reed-Solomon codes. By Lemma 4.1 below, the Berlekamp-Massey algorithm [Mas69] solves this problem in time quadratic in t.

Lemma 4.1 ([Mas69])**.** *Given the first $2t$ time-domain samples of a t-frequency sparse signal, Algorithm* BERLEKAMPMASSEY *finds the error locator polynomial $\Lambda(z)$ (given by Equation 2) in time $O(t^2)$.*

Proof. As shown in [AT08], finding the positions of the nonzero frequencies is equivalent to a generalization of the Reed-Solomon decoding problem to the complex field; then, solving this complex-field Reed-Solomon problem reduces to recovering the lowest-order linear recurrence ($\Lambda(z)$) that generates a given sequence of "syndromes" (which equal the x_i in our case) [Var97]. The running time is $O(t^2)$ where t is the degree of the polynomial [Mas69]. □

- PAN: By the definition of the error locator polynomial (given by Equation 2), its roots determine the set $\{f_1, \cdots, f_t\}$ of nonzero frequencies of x. Thus, we can use the Pan root-finding algorithm [Pan02] to find the complex roots of $\Lambda(z)$.

  Lemma 4.2. *([Pan02]) For a polynomial $p(z)$ of degree t with complex coefficients and whose complex roots are located in the unit disk $\{z : |z| \leq 1\}$, the* PAN *algorithm approximates all the roots of $p(z)$ with an absolute error of at most 2^{-b} and using $O(t \log^2 t(\log^2 t + \log b))$ arithmetic operations on b-bit numbers, assuming that $b \geq t \log t$.*

- VANDERMONDE: Given the nonzero frequencies of x, the problem reduces to solving a system of linear equations in the values of those frequencies. The coefficient matrix of this system is a Vandermonde matrix. Thus, this system has the form:

$$V \begin{bmatrix} v_1 \\ \vdots \\ v_t \end{bmatrix} = s^{(T)} \text{ where } V_{i,j} = (\omega')^{if_j} \tag{3}$$

- INVERSEVANDERMONDE: The Vandermonde matrix can be inverted using the optimal algorithm given in [Zip90] to find the values of the nonzero coordinates of $\widehat{x}$.

  Lemma 4.3. *([Zip90]) Given a $t \times t$ Vandermonde matrix V, its inverse can be computed in time $O(t^2)$.*

Analysis of 1DSFFT

Theorem 4.4. *If x is a t-sparse signal of size n where $t \leq O(\log n)$, then on input $\{x_0, \cdots, x_{2t-1}\}$ the procedure* SIGNALFROMSYNDROME *(and hence the algorithm* 1DSFFT *) computes the spectrum $\widehat{x}$ in time $O(t^2 + t(\log \log n)^{O(1)})$.*

Proof. By Lemma 4.1, the time needed to construct the error locator polynomial $\Lambda(z)$ is $O(t^2)$. By Lemma 4.2 and since $t \leq \log n$, running the PAN algorithm with $b = \log n \log \log n$ requires $O(t \log^2 t(\log^2 t + \log \log n))$ arithmetic operations on $(\log n \log \log n)$-bit numbers. Since a $(\log n \log \log n)$-bit arithmetic operation can be implemented using $O((\log \log n)^{O(1)}) \log n$-bit operations and since we are assuming that arithmetic operations on $\log n$ bits can be performed in constant time, the total running time of the PAN algorithm is $O(t(\log \log n)^{O(1)})$. Note that since the roots of $\Lambda(z)$ are n-th roots (or later $\sqrt{n}$-roots) of unity, the precision of the algorithm is sufficient. Noting that, by Lemma 4.3, inverting the Vandermonde matrix requires $O(t^2)$ time completes the proof of the Theorem. $\qquad\square$

Description and Analysis of EXACT1DSFFT   The algorithm EXACT1DSFFT(Algorithm 4.2) computes the 1D spectrum with high probability over a random k-sparse input for any k. It uses $O(k)$ samples and runs in time $O(k(\log k + (\log \log n)^{O(1)}))$. The idea is the following: We fold the spectrum into $\theta(k/\log k)$ bins using the 1D version of the comb filter (cf. Section §2). As shown in Lemma 4.5, with high probability, each of the bins has $O(\log k)$ nonzero frequencies. In this case, SIGNALFROMSYNDROME$(n, \widehat{u}_j^{(T)})$ will then recover the original spectrum values. The generalization of this algorithm to the 2D case can be found in Section 4.2.

---

procedure EXACT1DSFFT(x, k)
    $\widehat{x} \leftarrow 0$
    $B \leftarrow \Theta(k/\log k)$ ▷ Such that $B \mid n$
    $T \leftarrow [2C \log k]$ for a sufficiently large constant C.
    for $\tau \in T$ **do**
        $u_i^{(\tau)} := x_{i(n/B)+\tau}$ for $i \in [B]$
        Compute $\widehat{u}^{(\tau)}$, the DFT of $u^{(\tau)}$
    end for
    for $j \in [B]$ **do**   ▷ $\widehat{u}_j^{(T)} = \{\widehat{u}_j^{(\tau)} : \tau \in T\}$.
        $\{(v_i, f_i)\}_{i \in [C \log k]} \leftarrow$ SIGNALFROMSYNDROME$(n, \widehat{u}_j^{(T)})$
        $\widehat{x}_{f_i} \leftarrow v_i$ for all $i \in [C \log k]$.
    end for
    return $\widehat{x}$
end procedure

Algorithm 4.2: Exact 1D sparse FFT algorithm for any sparsity k

---

Lemma 4.5. *Assume that $\widehat{x}$ is distributed according to the Bernoulli model of Section §2. If we fold the spectrum into $B = \theta(k/\log k)$ bins, then for a sufficiently large constant C, the probability that there is a bin with more than $C \log k$ nonzero frequencies is smaller than $O((1/k)^{0.5C \log C})$.*

Proof. The probability $\Pr(B, k, m)$ that there is a bin with more than m nonzero frequencies is bounded by:

$$\Pr(B, k, m) \leq B \binom{n/B}{m} \left(\frac{k}{n}\right)^m \leq B \left(\frac{ek}{Bm}\right)^m$$

Since $B = dk/\log k$ (for some constant $d > 0$), $m = C \log k$, we get:

$$\Pr(B, k, m) \leq B \left(\frac{ek}{Bm}\right)^m \leq \frac{dk}{\log k} \left(\frac{ed}{C}\right)^{C \log k} = O\left(\frac{1}{k^{0.5C \log C}}\right)$$

Theorem 4.6. *If $\widehat{x}$ is distributed according to the Bernoulli model of Section §2, then Algorithm* EX-ACT1DSFFT *runs in time $O(k(\log k + (\log \log n)^{O(1)}))$, uses $O(k)$ samples and returns the correct spectrum $\widehat{x}$ with probability at least $1 - O((1/k)^{0.5C \log C})$.*

Proof. By Lemma 4.5, with probability at least $1 - O((1/k)^{0.5C \log C})$, all the bins have at most $C \log k$ nonzero frequencies each. Then, Theorem 4.4 guarantees the success of SIGNALFROMSYNDROME$(n, \widehat{u}_j^{(T)})$ for every $j \in [B]$. This proves the correctness of EXACT1DSFFT.

The running time of the for loop over τ is $O(k \log k)$. By Theorem 4.4, SIGNALFROMSYNDROME$(n, \widehat{u}_j^{(T)})$ takes time $O(\log^2 k + \log k (\log \log n)^{O(1)})$ for every $j \in [B]$. Thus, the total running time of EX-ACT1DSFFT is $O(k(\log k + (\log \log n)^{O(1)}))$. For every $\tau \in [C \log k]$, computing $\widehat{u}^{(\tau)}$ requires $B = k/\log k$ samples. Thus, the total number of samples needed is $O(k)$. □

4.2 Exact 2D Algorithm for $k = O(n)$

Here, we generalize the EXACT1DSFFT to the 2D case. For $k/\log k \geq \sqrt{n}$, the generalization is straightforward and can be found in Alg. 4.3: EXACT2DSFFT1. For $k/\log k \leq \sqrt{n}$, the generaliztion requires an extra step and can be found in Alg. 4.4 : EXACT2DSFFT2.

When $k/\log k \geq \sqrt{n}$, the desired bucket size n/B is less than $\sqrt{n}$, so we can have one-dimensional buckets and recover the locations with a single application of syndrome decoding. When $k/\log k < \sqrt{n}$, we need to have two dimensional buckets to make them large enough. But this means syndrome decoding will not uniquely identify the locations, and we will need multiple tests.

---

procedure EXACT2DSFFT1(x, k)
    $B_1 \leftarrow \sqrt{n}$.
    $B_2 \leftarrow \theta(k/(\log k \sqrt{n}))$.
    $T \leftarrow [2C \log k]$ for a sufficiently large constant C.
    for $\tau \in T$ **do**
        Define $u_{i,j}^{(\tau)} := x_{i, j(\sqrt{n}/B_2) + \tau}$ for $(i, j) \in [B_1] \times [B_2]$.
        Compute the 2D FFT $\widehat{u}^{(\tau)}$ of $u^{(\tau)}$
    end for
    $\widehat{x} \leftarrow 0$.
    for $(i, j) \in [B_1] \times [B_2]$ **do** $\triangleright \widehat{u}_{i,j}^{(T)} := \{\widehat{u}_{i,j}^{(\tau)} : \tau \in T\}$.
        $\{(f_l, v_l)\}_{l \in [C \log k]} \leftarrow$ SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_{i,j}^{(T)})$
        $\widehat{x}_{i, f_l} \leftarrow v_l$ for all $l \in [C \log k]$.
    end for
    return $\widehat{x}$
end procedure

Algorithm 4.3: Exact 2D sparse FFT algorithm for sparsity $k/\log k \geq \sqrt{n}$

Description and Analysis of EXACT2DSFFT1 The algorithm EXACT2DSFFT1 applies to the case where
$k/\log k \geq \sqrt{n}$. As in the 1D case, we use $B = \Theta(k/\log k)$ buckets each of which having $\Theta(n \log k/k)$

frequencies mapping to it. We construct the buckets corresponding to a phase shift of τ along the second dimension for all $\tau \in [2C \log k]$. As in the 1D case, with high probability, each of those buckets will have at most $C \log k$ nonzero frequencies. The particular choice of the buckets above will ensure that the inputs to the SIGNALFROMSYNDROME procedure have the appropriate "syndrome" form.

Theorem 4.7. *If x is a k-sparse signal (with $k/\log k \geq \sqrt{n}$) distributed according to the Bernoulli model of Section §2, then Algorithm EXACT2DSFFT1 runs in time $O(k \log k)$, uses $O(k)$ samples and recovers the spectrum $\widehat{x}$ of x with probability at least $1 - O\left(1/k^{0.5C \log C}\right)$.*

Proof. For every $(i,j) \in [B_1] \times [B_2]$ and every $\tau \in [2C \log k]$, $\widehat{u}_{i,j}^{(\tau)} = \sum_{f_2 \equiv j \bmod B_2} \widehat{x}_{i,f_2} \omega^{-\tau f_2}$. Using

the same argument as in Lemma 4.5, with high probability, every bin $\widehat{u}_{i,j}^{(\tau)}$ has at most $C \log k$ nonzero frequencies. Noting that the function SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_{i,j}^{(T)})$ succeeds whenever $\widehat{u}_{i,j}^{(T)}$ are the syndromes of a $C \log k$-sparse signal, implies the correctness of EXACT2DSFFT1.

Computing $\widehat{u}^{(\tau)}$ for all $\tau \in T$ takes time $O(k \log k)$. Each call to SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_{i,j}^{(T)})$ takes time $O(\log^2 k + \log k (\log \log n)^{O(1)})$ by Theorem 4.4. Thus, the overall running time is $O(k(\log k + (\log \log n)^{O(1)})) = O(k \log k)$. For every $\tau \in [C \log k]$, computing $\widehat{u}^{(\tau)}$ requires $B_1 \times B_2 = k/\log k$ samples. Thus, the total number of samples needed is $O(k)$. $\square$

---

procedure EXACT2DSFFT2(x, k)
  $B \leftarrow \theta(k/\log k)$.
  $T \leftarrow \{0, 1, \cdots, 2C \log k - 1\}$ for a sufficiently large constant C.
  for $(\tau, s) \in T \times [4]$ **do**
      Compute the FFT $\widehat{u}^{\tau,s}$ of $u^{\tau,s}$ where for every $i \in [B]$, $u_i^{(\tau_1, \tau_2)} = x_{\tau_1, i(\sqrt{n}/B) + \tau_2}$
  end for
  $\widehat{x} \leftarrow 0$.
  for $i \in [B]$ **do**
      for $s \in [4]$ **do** $\triangleright \widehat{u}_i^{T,s} = \{\widehat{u}_i^{\tau,s} : \tau \in T\}$.
          $\{(f_l^{(s)}, v_l^{(s)})\}_{l \in [C \log k]} \leftarrow$ SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_i^{T,s})$
      end for
      $(\{f_0, \cdots, f_{O(\log k)}\}, \{y_0^{(s)}, \cdots, y_{O(\log k)}^{(s)}\}_{s \in [4]}) \leftarrow$ MATCH$(\{(f_l^{(s)}, v_l^{(s)})\}_{s \in [4], \, l \in [C \log k]})$
      for $l \in [O(\log k)]$ **do** $\triangleright y_l^{(S)} := \{y_l^{(s)} : s \in S = [4]\}$.
          $\{(g_0, w_0), (g_1, w_1)\} \leftarrow$ SIGNALFROMSYNDROME$(\sqrt{n}, y_l^{(S)})$
          $\widehat{x}_{f_l, g_j} \leftarrow w_j$ for all $j \in [2]$.
      end for
  end for
  return $\widehat{x}$
end procedure

Algorithm 4.4: Exact 2D sparse FFT algorithm for sparsity $k/\log k \leq \sqrt{n}$

Description and Analysis of EXACT2DSFFT2   The algorithm EXACT2DSFFT2 above applies to the case where $k/\log k \leq \sqrt{n}$. As in the 1D case, we use $B = \Theta(k/\log k)$ buckets, each of which having

$\Theta(n \log k / k)$ frequencies mapping to it (i.e. $\Theta(\sqrt{n} \log k / k)$ columns). We construct 4 sets of buckets (as opposed to 1 set in the 1D case). Those sets correspond to the phase shifts $(\tau, 0)$, $(\tau, 1)$, $(\tau, 2)$ and $(\tau, 3)$ for all $\tau \in [2C \log k]$. We run the SIGNALFROMSYNDROME procedure on each of those 4 sets. As opposed to the 1D case, the resulting values can be the superposition of 2 or more nonzero frequency components. However, as shown in Lemma 4.9, with high probability, all the obtained values correspond to the superposition of at most 2 nonzero frequency components. The 4 corresponding superpositions (one from each of the 4 sets) are then combined (by the MATCH procedure) to get the union $\{f_0, \cdots, f_{O(\log k)}\}$ of the sets $\{f_0^{(s)}, \cdots, f_{C \log k - 1}^{(s)}\}$ for all $s \in [4]$ along with the associated values $\{y_0^{(s)}, \cdots, y_{O(\log k)}^{(s)}\}$ (with a value 0 if the frequency did not appear for some s). Then, we give the 4 resulting superpositions as inputs to the SIGNALFROMSYNDROME procedure again. The particular choice of the 4 sets of buckets above ensures that those inputs have the appropriate "syndrome" form. The output of this procedure will then consist of original spectrum values.

Lemma 4.8. *With probability at least* $1 - O\left(1/k^{0.5C \log C}\right)$, *for every* $i \in [B]$ *and* $s \in [4]$, *the output of* SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_i^{T,s})$ *consists of all nonzero values of the form* $\displaystyle\sum_{f_2 \equiv i \bmod B} \widehat{x}_{f_1, f_2} \omega^{-s f_2}$ *for some* $f_1 \in [\sqrt{n}]$.

Proof. As in Lemma 4.5, we have that the probability that there is a bin with more than $|T|/2 = C \log k$ nonzero frequencies is at most $O\left(1/k^{0.5C \log C}\right)$. Moreover, for every $\tau \in T$, $i \in [B]$ and $s \in [4]$, we have:

$$\widehat{u}_i^{\tau, s} = \sum_{f_1 \in [\sqrt{n}]} \sum_{f_2 \equiv i \bmod B} \widehat{x}_{f_1, f_2} \omega^{-\tau f_1 - s f_2} \tag{4}$$

$$= \sum_{f_1 \in [\sqrt{n}]} \Big(\sum_{f_2 \equiv i \bmod B} \widehat{x}_{f_1, f_2} \omega^{-s f_2} \Big) \omega^{-\tau f_1} \tag{5}$$

Noting that the function SIGNALFROMSYNDROME$(\sqrt{n}, \widehat{u}_i^{T,s})$ succeeds whenever $\widehat{u}_i^{T,s}$ are the syndromes of a $|T|/2$-sparse signal, we get the desired statement. $\square$

Lemma 4.9. *The probability that there are more than 2 nonzero frequency components that superimpose in a power of* ω *(i.e., as in Equation (4),* $\widehat{x}_{f_1, f_2}$ *and* $\widehat{x}_{f_1', f_2'}$ *superimpose if* $f_1 = f_1'$ *and* $f_2 \equiv f_2' \bmod B$) *is at most* $O(\frac{k \log^2 k}{n})$.

Proof. Since $\sqrt{n}/B = \Theta(\sqrt{n} \log k / k)$ frequencies map to each power of ω in each bucket, the probability is upper bounded by

$$n \cdot k/n \cdot \binom{\sqrt{n}/B}{2} \left(\frac{k}{n}\right)^2 \le \frac{k^3}{2nB^2} = \Theta\left(\frac{k \log^2 k}{n}\right)$$

$\square$

Lemma 4.10. *With probability at least*

$$1 - O\left(k \log^2 k / n - 1/k^{0.5C \log C}\right),$$

for all $i \in [B]$ *the outputs of* SIGNALFROMSYNDROME$(\sqrt{n}, y_l^S)$ *for all* $l \in [C \log k]$ *consist of all nonzero* $\widehat{x}_{f_1, f_2}$ *where* $f_2 \equiv i \bmod B$.

Proof. By Lemmas 4.9 and 4.8, the probability that all bins have at most $C \log k$ nonzero frequencies and all powers of ω have at most 2 nonzero frequencies is at least $1 - O\left(k \log^2 k/n - 1/k^{0.5C \log C}\right)$. Then for every $i \in [B]$ and $s \in S = [4]$, there are at most $C \log k$ nonzero values of the form $\sum_{f_2 \equiv i \bmod B} \widehat{x}_{f_1, f_2} \omega^{-s f_2}$

where $f_1 \in [\sqrt{n}]$, and each of those sums consists of at most 2 terms. Thus, $y_l^{(S)}$ are the syndromes of a 2-sparse signal of the form $\widehat{x}_{f_l, g_0} \omega^{-g_0 r} + \widehat{x}_{f_l, g_1} \omega^{-g_1 r}$ where r is the time-domain index. This yields the desired statement. □

Theorem 4.11. *If x is a k-sparse signal (with $k/\log k \leq \sqrt{n}$) distributed according to the Bernoulli model of Section §2, then Algorithm* EXACT2DSFFT2 *runs in time $O(k(\log k + (\log \log n)^{O(1)}))$, uses $O(k)$ samples and recovers the spectrum $\widehat{x}$ of x with probability at least $1 - k \log^2 k/n - O\left(1/k^{0.5C \log C}\right)$.*

Proof. Lemma 4.10 implies that Algorithm EXACT2DSFFT2 succeeds with the desired probability.

Computing $\widehat{u}^{\tau, s}$ for all $\tau \in T$ and all $s \in [4]$ takes time $O(k \log k)$. The running time of the MATCH procedure is $O(\log k)$. By Theorem 4.4, each call to SIGNALFROMSYNDROME in the for loop over s takes time $O(\log^2 k + \log k (\log \log n)^{O(1)})$ whereas each one in the for loop over l takes time $O((\log \log n)^{O(1)})$. Thus, the overall running time is $O(k(\log k + (\log \log n)^{O(1)}))$.

For every $\tau \in [C \log k]$, computing $\widehat{u}^{(\tau)}$ requires $B = k/\log k$ samples. Thus, the total number of samples needed is $O(k)$. □

5 Algorithm for Robust Recovery

5.1 Preliminaries

Following [CT06] we say that a matrix A satisfies a *restricted isometry property* (RIP) of order t with constant $\delta > 0$ if, for all t-sparse vectors y, we have $\|Ay\|_2^2/\|y\|_2^2 \in [1 - \delta, 1 + \delta]$.

Suppose all columns A_i of an $N \times M$ matrix A have unit norm. Let $\mu = \max_{i \neq j} |A_i \cdot A_j|$ be the *coherence* of A. It is folklore[6] that A satisfies the RIP of order t with the constant $\delta = (t - 1)\mu$.

Suppose that the matrix A is an $M \times N$ submatrix of the $N \times N$ Fourier matrix F, with each the M rows of A chosen uniformly at random from the rows of F. It is immediate from the Hoeffding bound that if $M = b\mu^2 \log(N/\gamma)$ for some large enough constant $b > 1$ then the matrix A has coherence at most μ with probability $1 - \gamma$. Thus, for $M = \Theta(t^2 \cdot t \log N)$, A satisfies the RIP of order t with constant $\delta = 0.5$ with probability $1 - 1/N^t$.

The algorithm appears in Algorithm 5.1.

5.2 Correctness of each stage of recovery

Lemma 5.1. *Consider the recovery of a column/row j in* ROBUSTESTIMATECOL, *where $\widehat{u}$ and $\widehat{v}$ are the results of* FOLDTOBINS *on $\widehat{x}$. Let $y \in \mathbb{C}^{\sqrt{n}}$ denote the jth column/row of $\widehat{x}$. Suppose y is drawn from a permutation invariant distribution $y = y^{head} + y^{residue} + y^{gauss}$, where $\min_{i \in \text{supp}(y^{head})} |y_i| \geq L$, $\|y^{residue}\|_1 < \epsilon L$, and y^{gauss} is drawn from the $\sqrt{n}$-dimensional normal distribution $N_\mathbb{C}(0, \sigma^2 I_{\sqrt{n}})$ with standard deviation $\sigma = \epsilon L/n^{1/4}$ in each coordinate on both real and imaginary axes. We do not require that y^{head}, $y^{residue}$, and y^{gauss} are independent except for the permutation invariance of their sum.*

Consider the following bad events:

---

[6]It is a direct corollary of Gershgorin's theorem applied to any t columns of A.

```
procedure ROBUSTESTIMATECOL($\widehat{u}$, $\widehat{v}$, $T$, $T'$, IsCol, $J$, Ranks)
    $\widehat{w} \leftarrow 0$.
    $S \leftarrow \{\}$                                                    ▷ Set of changes, to be tested next round.
    for $j \in J$ do
        continue if Ranks[(IsCol, $j$)] $\geq \log \log n$.
        $i \leftarrow$ HIKPLOCATESIGNAL($\widehat{u}^{(T')}$, $T'$)          ▷ Procedure from [HIKP12a]: $O(\log^2 n)$ time
        $a \leftarrow \text{median}_{\tau \in T}\, \widehat{u}_j^\tau \omega^{\tau i}$.
        continue if $|a| < L/2$                                              ▷ Nothing significant recovered
        continue if $\sum_{\tau \in T} |\widehat{u}_j^\tau - a\omega^{-\tau i}|^2 \geq L^2 |T| / 10$   ▷ Bad recovery: probably not 1-sparse
        $b \leftarrow \text{mean}_{\tau \in T}\, \widehat{u}_j^\tau \omega^{\tau i}$.
        if IsCol then                                                        ▷ whether decoding column or row
            $\widehat{w}_{i,j} \leftarrow b$.
        else
            $\widehat{w}_{j,i} \leftarrow b$.
        end if
        $S \leftarrow S \cup \{i\}$.
        Ranks[$(1 - \text{IsCol}, i)$] += Ranks[(IsCol, $j$)].
        for $\tau \in T \cup T'$ do
            $\widehat{u}_j^{(\tau)} \leftarrow \widehat{u}_j^{(\tau)} - b\omega^{-\tau i}$
            $\widehat{v}_i^{(\tau)} \leftarrow \widehat{v}_i^{(\tau)} - b\omega^{-\tau i}$
        end for
    end for
    return $\widehat{w}$, $\widehat{u}$, $\widehat{v}$, $S$
end procedure
procedure ROBUST2DSFFT($x$, $k$)
    $T, T' \subset [\sqrt{n}], |T| = |T'| = O(\log n)$
    for $\tau \in T \cup T'$ do
        $\widehat{u}^{(\tau)} \leftarrow$ FOLDTOBINS($x, \sqrt{n}, 1, 0, \tau$).
        $\widehat{v}^{(\tau)} \leftarrow$ FOLDTOBINS($x, 1, \sqrt{n}, \tau, 0$).
    end for
    $\widehat{z} \leftarrow 0$
    Ranks $\leftarrow 1^{[2] \times [\sqrt{n}]}$                             ▷ Rank of vertex (iscolumn, index)
    $S_{col} \leftarrow [\sqrt{n}]$                                          ▷ Which columns to test
    for $t \in [C \log n]$ do
        $\{\widehat{w}, \widehat{u}, \widehat{v}, S_{row}\} \leftarrow$ ROBUSTESTIMATECOL($\widehat{u}, \widehat{v}, T, T'$, true, $S_{col}$, Ranks).
        $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
        $S_{row} \leftarrow [\sqrt{n}]$ if $t = 0$                           ▷ Try every row the first time
        $\{\widehat{w}, \widehat{v}, \widehat{u}, S_{col}\} \leftarrow$ ROBUSTESTIMATECOL($\widehat{v}, \widehat{u}, T, T'$ false, $S_{row}$, Ranks).
        $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
    end for
    return $\widehat{z}$
end procedure
```

Algorithm 5.1: Robust 2D sparse FFT algorithm for $k = \Theta(\sqrt{n})$

- *False negative:* $\operatorname{supp}(y^{head}) = \{i\}$ *and* ROBUSTESTIMATECOL *does not update coordinate* i.

- *False positive:* ROBUSTESTIMATECOL *updates some coordinate* i *but* $\operatorname{supp}(y^{head}) \neq \{i\}$.

- *Bad update:* $\operatorname{supp}(y^{head}) = \{i\}$ *and coordinate* i *is estimated by* b *with* $\left|b - y_i^{head}\right| > \|y^{residue}\|_1 + \sqrt{\frac{\log \log n}{\log n}} \epsilon L$.

For any constant c and ϵ below a sufficiently small constant, there exists a distribution over sets T, T' of size $O(\log n)$, such that as a distribution over y and T, T' we have

- *The probability of a false negative is* $1/\log^c n$.

- *The probability of a false positive is* $1/n^c$.

- *The probability of a bad update is* $1/\log^c n$.

Proof. Let $\check{y}$ denote the 1-dimensional inverse DFT of y. Note that

$$\widehat{u}_j^{(\tau)} = \check{y}_\tau$$

by definition. Therefore, the goal of ROBUSTESTIMATECOL is simply to perform reliable 1-sparse recovery with $O(\log n)$ queries. Fortunately, [HIKP12a] solved basically the same problem, although with more false positives than we want here.

We choose T' according to the LOCATEINNER procedure from [HIKP12a]; the set T is chosen uniformly at random from $[\sqrt{n}]$. We have that

$$\widehat{u}_j^{(\tau)} = \sum_{i \in [\sqrt{n}]} y_i \omega^{-\tau i}.$$

This is exactly what the procedure HASHTOBINS of [HIKP12a] approximates up to a small error term. Therefore, the same analysis goes through (Lemma 4.5 of [HIKP12a]) to get that HIKPLOCATESIGNAL returns i with $1 - 1/\log^c n$ probability if $|y_i| \geq \|y_{-i}\|_2$, where we define $y_{-i} := y_{[\sqrt{n}] \setminus \{i\}}$.

Define $A \in \mathbb{C}^{|T| \times \sqrt{n}}$ to be the rows of the inverse Fourier matrix indexed by T, normalized so $|A_{i,j}| = 1$. Then $\widehat{u}_j^{(\tau)} = (Ay)_\tau$.

First, we prove

$$\|y^{residue} + y^{gauss}\|_2 = O(\epsilon L) \tag{6}$$

with all but n^{-c} probability. We have that $\mathbb{E}[\|y^{gauss}\|_2^2] = 2\epsilon^2 L^2$, so $\|y^{gauss}\|_2 \leq 3\epsilon L$ with all but $e^{-\Omega(\sqrt{n})} < 1/n^c$ probability by concentration of chi-square variables. We also have that $\|y^{residue}\|_2 \leq \|y^{residue}\|_1 \leq \epsilon L$.

Next, we show

$$\|A(y^{residue} + y^{gauss})\|_2 = O(\epsilon L \sqrt{|T|}) \tag{7}$$

with all but n^{-c} probability. We have that Ay^{gauss} is drawn from $N_{\mathbb{C}}(0, \epsilon^2 L^2 I_{|T|})$ by the rotation invariance of Gaussians, so

$$\|Ay^{gauss}\|_2 \leq 3\epsilon L \sqrt{|T|} \tag{8}$$

with all but $e^{-\Omega(|T|)} < n^{-c}$ probability. Furthermore, A has entries of magnitude 1 so $\|Ay^{residue}\|_2 \le \|y^{residue}\|_1 \sqrt{|T|} = \epsilon L \sqrt{|T|}$.

Consider the case where $\mathrm{supp}(y^{head}) = \{i\}$. From Equation (6) we have

$$\|y_{-i}\|_2^2 \le \|y^{gauss} + y^{residue}\|_2^2 \le O(\epsilon^2 L^2) < L^2 \le \|y_i\|_2^2 \tag{9}$$

so i is located with $1 - 1/\log^c n$ probability by HIKPLOCATESIGNAL.

Next, we note that for any i, as a distribution over $\tau \in [\sqrt{n}]$,

$$\mathbb{E}_\tau\left[\left|\widehat{u}_j^{(\tau)} - y_i\omega^{-\tau i}\right|^2\right] = \|y_{-i}\|_2^2$$

and so (analogously to Lemma 4.6 of [HIKP12a], and for any i), since $a = \mathrm{median}_{\tau \in T}\, \widehat{u}_j^{(\tau)}\omega^{\tau i}$ we have

$$|a - y_i|^2 \le 5\|y_{-i}\|_2^2 \tag{10}$$

with probability $1 - e^{-\Omega(|T|)} = 1 - 1/n^c$ for some constant c. Hence if $\{i\} = \mathrm{supp}(y^{head})$, we have $|a - y_i|^2 \le O(\epsilon^2 L^2)$ and therefore $|a| > L/2$, passing the first check on whether i is valid.

For the other check, we have that with $1 - 1/n^c$ probability

$$\begin{aligned}
(\sum_{\tau \in T}\left|\widehat{u}_j^{(\tau)} - a\omega^{-\tau i}\right|^2)^{1/2} &= \|A(y - ae_i)\|_2 \\
&\le \|A(y^{gauss} + y^{residue} + (y_i^{head} - a)e_i)\|_2 \\
&\le \|A(y^{gauss} + y^{residue})\|_2 + \left|y_i^{head} - a\right|\sqrt{|T|} \\
&\le \|A(y^{gauss} + y^{residue})\|_2 + (\left|y_i^{residue} + y_i^{gauss}\right| + |y_i - a|)\sqrt{|T|} \\
&\le O(\epsilon L \sqrt{|T|}).
\end{aligned}$$

where the last step uses Equation 7. This gives

$$\sum_{\tau \in T}\left|\widehat{u}_j^{(\tau)} - a\omega^{-\tau i}\right|^2 = O(\epsilon^2 L^2 |T|) < L^2 |T|/10$$

so the true coordinate i passes both checks. Hence the probability of a false negative is $1/\log^c n$ as desired.

Now we bound the probability of a false positive. First consider what happens to any other coordinate $i' \ne i$ when $\left|\mathrm{supp}(y^{head})\right| = \{i\}$. We get some estimate a' of its value. Since $A/\sqrt{|T|}$ satisfies an RIP of order 2 and constant $1/4$, by the triangle inequality and Equation 7 we have that with $1 - n^{-c}$ probability,

$$\begin{aligned}
\|A(y - a'e_{i'})\|_2 &\ge \|A(y_i^{head}e_i - a'e_{i'})\|_2 - \|A(y^{gauss} + y^{residue})\|_2 \\
&\ge y_i^{head}\sqrt{|T|} \cdot (3/4) - O(\epsilon L \sqrt{|T|}) \\
&> L\sqrt{|T|}/2.
\end{aligned}$$

Hence the second condition will be violated, and i' will not pass. Thus if $\left|\mathrm{supp}(y^{head})\right| = 1$, the probability of a false positive is at most n^{-c}.

Next, consider what happens to the result i of HIKPLOCATESIGNAL when $\left|\mathrm{supp}(y^{head})\right| = 0$. From Equation (6) and Equation (7) we have that with $1 - n^{-c}$ probability:

$$|a - y_i|^2 \le 5\|y_{-i}\|_2^2 \le O(\epsilon^2 L^2).$$

Therefore, from Equation 6,

$$|a| \leq |y_i| + |a - y_i| \leq \|y^{residue} + y^{gauss}\|_2 + |a - y_i| = O(\epsilon L) < L/2$$

so the first check is not passed and i is not recovered.

Now suppose $|\text{supp}(y_{head})| > 1$. Lemma 5.2 says that with $1 - n^{-c}$ probability over the permutation, no (i, a) satisfies

$$\|A(y^{head} - ae_i)\|_2^2 < L^2 |T| / 5.$$

But then, from Equation 8

$$\begin{aligned}
\|A(y - ae_i)\|_2 &\geq \|A(y^{head} - ae_i)\|_2 - \|Ay^{gauss}\|_2 \\
&> L\sqrt{|T|/5} - O(\epsilon L \sqrt{|T|}) \\
&> L\sqrt{|T|/10}
\end{aligned}$$

so no i will pass the second check. Thus the probability of a false positive is $1/n^c$.

Finally, consider the probability of a bad update. We have that

$$b = \underset{\tau \in T}{\text{mean}}(Ay)_\tau \omega^{\tau i} = y_i^{head} + \underset{\tau \in T}{\text{mean}}(Ay^{residue} + Ay^{gauss})_\tau \omega^{\tau i}$$

and so

$$\left| b - y_i^{head} \right| \leq \left| \underset{\tau \in T}{\text{mean}}(Ay^{residue})_\tau \omega^{\tau i} \right| + \left| \underset{\tau \in T}{\text{mean}}(Ay^{gauss})_\tau \omega^{\tau i} \right|.$$

We have that

$$\left| \underset{\tau \in T}{\text{mean}}(Ay^{residue})_\tau \omega^{\tau i} \right| \leq \underset{\tau \in T}{\max} \left| (Ay^{residue})_\tau \right| \leq \|y^{residue}\|_1$$

.

We know that Ay^{gauss} is $N_{\mathbb{C}}(0, \epsilon^2 L^2 I_{|T|})$. Hence its mean is a complex Gaussian with standard deviation $\epsilon L/\sqrt{|T|}$ in both the real and imaginary axes. This means the probability that

$$\left| b - y_i^{head} \right| > \|y^{residue}\|_1 + t\epsilon L/\sqrt{|T|}$$

is at most $e^{-\Omega(t^2)}$. Setting $t = \sqrt{\log \log^c n}$ gives a $1/\log^c n$ chance of a bad update, for sufficiently large $|T| = O(\log n)$. $\qquad\square$

The following is the robust analog of Lemma 3.3.

Lemma 5.2. *Let $y \in \mathbb{C}^m$ be drawn from a permutation invariant distribution with $r \geq 2$ nonzero values. Suppose that all the nonzero entries of y have absolute value at least L. Choose $T \subset [m]$ uniformly at random with $t := |T| = O(c^3 \log m)$*

Then, the probability that there exists a y' with $\|y'\|_0 \leq 1$ and

$$\|(\check{y} - \check{y}')_T\|_2^2 < \epsilon L^2 t/n$$

is at most $c^3 (\frac{c}{m-r})^{c-2}$ whenever $\epsilon < 1/8$.

Proof. Let $A = \sqrt{1/t} F_{T \times *}$ be $\sqrt{1/t}$ times the submatrix of the Fourier matrix with rows from T, so

$$\|(\check{y} - \check{y}')_T\|_2^2 = \|A(y - y')\|_2^2 t/n.$$

By a coherence bound (see Section 5.1), with $1 - 1/m^c$ probability A satisfies the RIP of order $2c$ with constant 0.5. We would like to bound

$$P := \Pr[\exists y' : \|A(y - y')\|_2^2 < \epsilon L^2 \text{ and } \|y'\|_0 \leq 1]$$

If $r \leq c - 1$, then $y - y'$ is c-sparse and

$$\begin{aligned} \|A(y - y')\|_2^2 &\geq \|y - y'\|_2^2/2 \\ &\geq (r - 1)L^2/2 \\ &> \epsilon L^2 \end{aligned}$$

as long as $\epsilon < 1/2$, giving $P = 0$. Henceforth, we can assume $r \geq c$. When drawing y, first place $r - (c - 1)$ coordinates into u then place the other $c - 1$ values into v, so that $y = u + v$. Condition on u, so v is a permutation distribution over $m - r + c - 1$ coordinates. We would like to bound

$$P = \Pr_v[\exists y' : \|A(u + v - y')\|_2^2 < \epsilon L^2 \text{ and } \|y'\|_0 \leq 1].$$

Let w be any c-sparse vector such that $\|A(u + w)\|_2^2 < \epsilon L^2$ (and note that if no such w exists, then since $v - y'$ is c-sparse, $P = 0$). Then recalling that for any norm $\|\cdot\|$, $\|a\|^2 \leq 2\|b\|^2 + 2\|a + b\|^2$ and hence $\|a + b\|^2 \geq \|a\|^2/2 - \|b\|^2$,

$$\begin{aligned} \|A(u + v - y')\|_2^2 &\geq \|A(v - y' - w)\|_2^2/2 - \|A(u + w)\|_2^2 \\ &\geq \|v - y' + w\|_2^2/4 - \epsilon L^2. \end{aligned}$$

Hence

$$P \leq \Pr_v[\exists y' : \|v - y' + w\|_2^2 < 8\epsilon L^2 \text{ and } \|y'\|_0 \leq 1].$$

Furthermore, we know that $\|v - y' + w\|_2^2 \geq L^2(|\text{supp}(v) \setminus \text{supp}(w)| - 1)$. Thus if $\epsilon < 1/8$,

$$\begin{aligned} P &\leq \Pr_v[|\text{supp}(v) \setminus \text{supp}(w)| \leq 1] \\ &\leq \frac{c + (m - r + c - 1)c(c - 1)/2}{\binom{m - r + c - 1}{c - 1}} \\ &< c^3 \left(\frac{c}{m - r}\right)^{c - 2} \end{aligned}$$

as desired. $\qquad \square$

5.3 Overall Recovery

Recall that we are considering the recovery of a signal $\widehat{x} = \widehat{x^*} + \widehat{w} \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, where $\widehat{x^*}$ is drawn from the Bernoulli model with expected $k = a\sqrt{n}$ nonzeros for a sufficiently small constant a, and $\widehat{w} \sim N_{\mathbb{C}}(0, \sigma^2 I_n)$ with $\sigma = \epsilon L \sqrt{k/n} = \Theta(\epsilon L/n^{1/4})$ for sufficiently small ϵ.

It will be useful to consider a bipartite graph representation G of $\widehat{x^*}$. We construct a bipartite graph with $\sqrt{n}$ nodes on each side, where the left side corresponds to rows and the right side corresponds to columns. For each $(i,j) \in \text{supp}(\widehat{x^*})$, we place an edge between left node i and right node j of weight $\widehat{x^*}_{(i,j)}$.

Our algorithm is a "peeling" procedure on this graph. It iterates over the vertices, and can with a "good probability" recover an edge if it is the only incident edge on a vertex. Once the algorithm recovers an edge, it can remove it from the graph. The algorithm will look at the column vertices, then the row vertices, then repeat; these are referred to as *stages*. Supposing that the algorithm succeeds at recovery on each vertex, this gives a canonical order to the removal of edges. Call this the *ideal* ordering.

In the ideal ordering, an edge e is removed based on one of its incident vertices v. This happens after all other edges reachable from v without passing through e are removed. Define the *rank* of v to be the number of such reachable edges, and $\text{rank}(e) = \text{rank}(v) + 1$ (with $\text{rank}(v)$ undefined if v is not used for recovery of any edge).

Lemma 5.3. *Let c, α be arbitrary constants, and a a sufficiently small constant depending on c, α. Then with $1 - \alpha$ probability every component in G is a tree and at most $k/\log^c n$ edges have rank at least $\log \log n$.*

Proof. Each edge of G appears independently with probability $k/n = a/\sqrt{n}$. There are at most $\sqrt{n}^t$ cycles of length t. Hence the probability that any cycle of length t exists is at most a^t, so the chance any cycle exists is less than $a^2/(1 - a^2) < \alpha/2$ for sufficiently small a.

Each vertex has expected degree $a < 1$. Exploring the component for any vertex v is then a subcritical branching process, so the probability that v's component has size at least $\log \log n$ is $1/\log^c n$ for sufficiently small a. Then for each edge, we know that removing it causes each of its two incident vertices to have component size less than $\log \log n - 1$ with $1 - 1/\log^c n$ probability. Since the rank is one more than the size of one of these components, the rank is less than $\log \log n$ with $1 - 2/\log^c n$ probability.

Therefore, the expected number of edges with rank at least $\log \log n$ is $2k/\log^c n$. Hence with probability $1 - \alpha/2$ there are at most $(1/\alpha)4k/\log^c n$ such edges; adjusting c gives the result. $\qquad\square$

Lemma 5.4. *Let* ROBUST2DSFFT' *be a modified* ROBUST2DSFFT *that avoids false negatives or bad updates: whenever a false negative or bad update would occur, an oracle corrects the algorithm. With large constant probability,* ROBUST2DSFFT' *recovers $\widehat{z}$ such that there exists a $(k/\log^c n)$-sparse $\widehat{z}'$ satisfying*

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \le 6\sigma^2 n.$$

Furthermore, only $O(k/\log^c n)$ false positives or bad updates are caught by the oracle.

Proof. One can choose the random $\widehat{x^*}$ by first selecting the topology of the graph G, and then selecting the random ordering of the columns and rows of the matrix. Note that reordering the vertices only affects the ideal ordering by a permutation within each stage of recovery; the set of edges recovered at each stage in the ideal ordering depends only on the topology of G. Suppose that the choice of the topology of the graph satisfies the thesis of Lemma 5.3 (which occurs with large constant probability). We will show that with large constant probability (over the space of random permutations of the rows and columns), ROBUST2DSFFT' follows the ideal ordering and the requirements of Lemma 5.1 are satisfied at every stage.

For a recovered edge e, we define the "residue" $\widehat{x^*}_e - \widehat{z}_e$. We will show that if e has rank r, then $\left|\widehat{x^*}_e - \widehat{z}_e\right| \le r\sqrt{\frac{\log \log n}{\log n}}\epsilon L$.

During attempted recovery at any vertex v during the ideal ordering (including attempts on vertices which do not have exactly one incident edge), let $y \in \mathbb{C}^{\sqrt{n}}$ be the associated column/row of $\widehat{x} - \widehat{z}$. We split y into three parts $y = y^{head} + y^{residue} + y^{gauss}$, where y^{head} contains the elements of $\widehat{x^*}$ not in $\text{supp}(\widehat{z})$,

$y^{residue}$ contains $\widehat{x^*} - \widehat{z}$ over the support of $\widehat{z}$, and y^{gauss} contains $\widehat{w}$ (all restricted to the column/row corresponding to v). Let $S = \text{supp}(y^{residue})$ contain the set of edges incident on v that have been recovered so far. We have by the inductive hypothesis that $\|y^{residue}\|_1 \leq \sum_{e \in S} \text{rank}(e) \sqrt{\frac{\log \log n}{\log n}} \epsilon L$. Since the algorithm verifies that $\sum_{e \in S} \text{rank}(e) \leq \log \log n$, we have

$$\|y^{residue}\|_1 \leq \sqrt{\frac{\log^3 \log n}{\log n}} \epsilon L < \epsilon L.$$

Furthermore, y is permutation invariant: if we condition on the values and permute the rows and columns of the matrix, the algorithm will consider the permuted y in the same stage of the algorithm.

Therefore the conditions for Lemma 5.1 hold. This means that the chance of a false positive is $1/n^c$, so by a union bound this never occurs. Because false negatives never occur by assumption, this means we continue following the ideal ordering. Because bad updates never occur, new residuals have magnitude at most

$$\|y^{residue}\|_1 + \sqrt{\frac{\log \log n}{\log n}} \epsilon L.$$

Because $\|y^{residue}\|_1 / \left(\sqrt{\frac{\log \log n}{\log n}} \epsilon L\right) \leq \sum_{e \in S} \text{rank}(e) = \text{rank}(v) = \text{rank}(e) - 1$, each new residual has magnitude at most

$$\text{rank}(e) \sqrt{\frac{\log \log n}{\log n}} \epsilon L \leq \epsilon L. \tag{11}$$

as needed to complete the induction.

Given that we follow the ideal ordering, we recover every edge of rank at most $\log \log n$. Furthermore, the residue on every edge we recover is at most ϵL. By Lemma 5.3 there are at most $k/\log^c n$ edges that we do not recover. From Equation (11), the squared ℓ_2 norm of the residues is at most $\epsilon^2 L^2 k = \epsilon^2 C^2 \sigma^2 n / k \cdot k < \sigma^2 n$ for ϵ small enough. Since $\|\widehat{w}\|_2^2 < 2\sigma^2 n$ with overwhelming probability, there exists a $\widehat{z}'$ so that

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \leq 2\|\widehat{z} - \widehat{x^*} - \widehat{z}'\|_2^2 + 2\|w\|_2^2 \leq 6\sigma^2 n.$$

Finally, we need to bound the number of times the oracle catches false positives or bad updates. The algorithm applies Lemma 5.1 only $2\sqrt{n} + O(k) = O(k)$ times. Each time has a $1/\log^c n$ chance of a false positive or bad update. Hence the expected number of false positives or bad updates is $O(k/\log^c n)$. $\qquad\square$

Lemma 5.5. *For any constant $\alpha > 0$, the algorithm* ROBUST2DSFFT *can with probability $1 - \alpha$ recover $\widehat{z}$ such that there exists a $(k/\log^{c-1} n)$-sparse $\widehat{z}'$ satisfying*

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \leq 6\sigma^2 n$$

using $O(k \log n)$ samples and $O(k \log^2 n)$ time.

Proof. To do this, we will show that changing the effect of a single call to ROBUSTESTIMATECOL can only affect $\log n$ positions in the output of ROBUST2DSFFT. By Lemma 5.4 we can, with large constant probability turn ROBUST2DSFFT into ROBUST2DSFFT' with only $O(k/\log^c n)$ changes to calls to RO-BUSTESTIMATECOL. This means the output of ROBUST2DSFFT and of ROBUST2DSFFT' only differ in $O(k/\log^{c-1} n)$ positions.

We view ROBUSTESTIMATECOL as trying to estimate a vertex. Modifying it can change from recovering one edge (or none) to recovering a different edge (or none). Thus, a change can only affect at most two calls to ROBUSTESTIMATECOL in the next stage. Hence in r stages, at most 2^{r-1} calls may be affected, so at most 2^r edges may be recovered differently.

Because we refuse to recover any edge with rank at least $\log \log n$, the algorithm has at most $\log \log n$ stages. Hence at most $\log n$ edges may be recovered differently as a result of a single change to ROBUSTESTIMATECOL. $\qquad\square$

Theorem 5.6. *Our overall algorithm can recover $\widehat{x}'$ satisfying*

$$\|\widehat{x} - \widehat{x}'\|_2^2 \leq 12\sigma^2 n + \|\widehat{x}\|_2^2/n^c$$

with probability $1 - \alpha$ for any constants $c, \alpha > 0$ in $O(k \log n)$ samples and $O(k \log^2 n)$ time, where $k = a\sqrt{n}$ for some constant $a > 0$.

Proof. By Lemma 5.5, we can recover an $O(k)$-sparse $\widehat{z}$ such that there exists an $(k/\log^{c-1} n)$-sparse $\widehat{z}'$ with

$$\|\widehat{x} - \widehat{z} - \widehat{z}'\|_2^2 \leq 6\sigma^2 n.$$

with arbitrarily large constant probability for any constant c using $O(k \log^2 n)$ time and $O(k \log n)$ samples. Then by Theorem B.1 in Appendix B, we can recover a $\widehat{z}'$ in $O(k \log^2 n)$ time and $O(k \log^{4-c} n)$ samples satisfying

$$\|\widehat{x} - \widehat{z} - \widehat{z}'\|_2^2 \leq 12\sigma^2 n + \|\widehat{x}\|_2^2/n^c$$

and hence $\widehat{x}' := \widehat{z} + \widehat{z}'$ is a good reconstruction for $\widehat{x}$. $\qquad\square$

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.

[AT08]   M. Akcakaya and V. Tarokh. A frame construction and a universal distortion bound for sparse representations. *Signal Processing, IEEE Transactions on*, 56(6):2443 –2450, june 2008.

[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.

[BK95]   V. Bahskarna and K. Konstantinides. *Image and video compression standards : algorithms and architectures*. Kluwer Academic Publishers, 1995.

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

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

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

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

[Gol99]   O. Goldreich. Modern cryptography, probabilistic proofs and pseudorandomness. *Algorithms and Combinatorics*, 17, 1999.

[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.

[HPN97]   B. G. Haskell, A. Puri, and A. N. Netravali. *Digital video : an introduction to MPEG-2*. Chapman and Hall, 1997.

[IGS07]   M. A. Iwen, A. Gilbert, and M. Strauss. Empirical evaluation of a sub-linear time sparse dft algorithm. *Communications in Mathematical Sciences*, 5, 2007.

[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.

[KS01]   A. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. Society for Industrial and Applied Mathematics, 2001.

[Lev93]   L.A. Levin. Randomness and non-determinism. *J. Symb. Logic*, 58(3):1102–1103, 1993.

[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.

[Mas69]   J. Massey. Shift-register synthesis and bch decoding. *Information Theory, IEEE Transactions on*, 15(1):122 – 127, jan 1969.

[MEH09]   Y. Matsuki, M. Eddy, and J. Herzfeld. Spectroscopy by integration of frequency and time domain information (sift) for fast acquisition of high resolution dark spectra. *J. Am. Chem. Soc.*, 2009.

[MS77]   F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, 1977.

[MV08]   M. Mitzenmacher and S. Vadhan. Why simple hash functions work: Exploiting the entropy in a data stream. *SODA*, 2008.

[Nis10]   D. Nishimura. *Principles of Magnetic Resonance Imaging*. Society for Industrial and, 2010.

[Pan02]   V. Y. Pan. Univariate polynomials: Nearly optimal algorithms for numerical factorization and root-finding. *J. Symbolic Computation*, 2002.

[PW11]   E. Price and D. P. Woodruff. $(1 + \epsilon)$-approximate sparse recovery. *FOCS*, 2011.

[Var97]   A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. *STOC*, 1997.

[Wal91]   G. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 1991.

[Zip90]   R. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375 – 403, 1990.

A   Sample lower bound for our distribution

We will show that the lower bound on ℓ_2/ℓ_2 recovery from [PW11] applies to our setting with a simple reduction. First, we state their bound:

Lemma A.1 ([PW11] section 4). *For any $k < n/\log n$ and constant $\epsilon > 0$, there exists a distribution D_k over k-sparse vectors in $\{0, 1, -1\}^n$ such that, for every distribution of matrices $A \in \mathbb{R}^{m \times n}$ with $m = o(k \log(n/k))$ and recovery algorithms $\mathcal{A}$,*

$$\Pr[\|\mathcal{A}(A(x + w)) - x\|_2 < \sqrt{k}/5] < 1/2$$

as a distribution over $x \sim D_k$ and $w \sim N(0, \sigma^2 I_n)$ with $\sigma^2 = \epsilon k/n$, as well as over A and $\mathcal{A}$.

First, we note that we can replace D_k with U_k, the uniform distribution over k-sparse vectors in $\{0, 1, -1\}^n$ in Lemma A.1. To see this, suppose we have an $(A, \mathcal{A})$ that works with $1/2$ probability over U_k. Then for any k-sparse $x \in \{0, 1, -1\}^n$, if we choose a random permutation matrix P and sign flip matrix S, $PSx \sim U_k$. Hence, the distribution of matrices APS and algorithm $\mathcal{A}'(x) = \mathcal{A}((PS)^{-1}x)$ works with $1/2$ probability for any x, and therefore on average over D_k. This implies that A has $\Omega(k \log(n/k))$ rows by Lemma A.1. Hence, we can set $D_k = U_k$ in Lemma A.1.

Our algorithm works with $3/4$ probability over vectors x that are not necessarily k-sparse, but have a binomial number $B(n, k/n)$ of nonzeros. That is, it works over the distribution U that is $U_{k'} : k' \sim B(n, k/n)$. With $1 - e^{-\Omega(k)} > 3/4$ probability, $k' \in [k/2, 2k]$. Hence, our algorithm works with at least $1/2$ probability over $(U_{k'} : k' \sim B(n, k/n) \cap k' \in [k/2, 2k])$. By an averaging argument, there must exist a $k' \in [k/2, 2k]$ where our algorithm works with at least $1/2$ probability over $U_{k'}$; but the lemma implies that it must therefore take $\Omega(k' \log(n/k')) = \Omega(k \log(n/k))$ samples.

B   Robust 2D FFTs

This section outlines the straightforward generalization of [HIKP12a] to two dimensions, as well as how to incorporate the extra parameter $\widehat{z}$ of already recovered coefficients. Relative to our result of Theorem 5.6,

this result takes more samples. However, it does not require that the input be from a random distribution and is used as a subroutine by Theorem 5.6 after decreasing the sparsity by a $\log^c n$ factor.

Because we use this as a subroutine after computing an estimate $\widehat{z}$ of $\widehat{x}$, we actually want to estimate $\widehat{x} - \widehat{z}$ where we have oracle access to x and to $\widehat{z}$.

Theorem B.1. *There is a variant of [HIKP12a] algorithm that will, given $x, \widehat{z} \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, return $\widehat{x'}$ with*

$$\|\widehat{x} - \widehat{z} - \widehat{x'}\|_2 \leq 2 \cdot \min_{k\text{-sparse } \widehat{x^*}} \|\widehat{x} - \widehat{z} - \widehat{x^*}\|_2^2 + \|\widehat{x}\|_2^2 / n^c$$

with probability $1 - \alpha$ for any constants $c, \alpha > 0$ in time

$$O(k \log(n/k) \log^2 n + |\text{supp}(\widehat{z})| \log(n/k) \log n),$$

using $O(k \log(n/k) \log^2 n)$ samples of x.

Proof. We need to modify [HIKP12a] in two ways: by extending it to two dimensions and by allowing the parameter $\widehat{z}$. We will start by describing the adaptation to two dimensions.

The basic idea of [HIKP12a] is to construct from Fourier measurements a way to "hash" the coordinates in $B = O(k)$ bins. There are three basic components that are needed: a *permutation* that gives nearly pairwise independent hashing to bins; a *filter* that allows for computing the sum of bins using Fourier measurements; and the *location* estimation needs to search in both axes. The permutation is the main subtlety.

Permutation Let $\mathcal{M} \subset [\sqrt{n}]^{2 \times 2}$ be the set of matrices with odd determinant. For notational purposes, for $v = (i, j)$ we define $x_v := x_{i,j}$.

Definition B.2. *For $M \in \mathcal{M}$ and $a, b \in [\sqrt{n}]^2$ we define the permutation $P_{M,a,b} \mathbb{C}^{\sqrt{n} \times \sqrt{n}} \to \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$ by*

$$(P_{M,a,b}x)_v = x_{M(v-a)} \omega^{v^T M b}.$$

We also define $\pi_{M,b}(v) = M(v - b) \mod \sqrt{n}$.

Claim B.3. $\widehat{P_{M,a,b}x}_{\pi_{M^T,b}(v)} = \widehat{x}_v \omega^{v^T M^T a}$

Proof.

$$\widehat{P_{M,a,b}x}_{M(v-b)} = \frac{1}{\sqrt{n}} \sum_{u \in [\sqrt{n}]^2} \omega^{u^T M(v-b)} (P_{M,a,b}x)_u$$

$$= \frac{1}{\sqrt{n}} \sum_{u \in [\sqrt{n}]^2} \omega^{u^T M(v-b)} x_{M(u-a)} \omega^{u^T M b}$$

$$= \omega^{v^T M^T a} \frac{1}{\sqrt{n}} \sum_{u \in [\sqrt{n}]^2} \omega^{v^T M^T (u-a)} x_{M(u-a)}$$

$$= \widehat{x}_i \omega^{v^T M^T a}$$

where we used that M^T is a bijection over $[\sqrt{n}]^2$ because $\det(M)$ is odd. $\qquad\square$

This gives a lemma analogous to Lemma 2.4 of [HIKP12a].

Lemma B.4. *Suppose $v \in [\sqrt{n}]^2$ is not 0. Then*

$$\Pr_{M \sim \mathcal{M}}[Mv \in [-C, C]^2 \pmod{\sqrt{n}}] \leq O(\frac{C^2}{n}).$$

Proof. For any u, define $G(u)$ to be the largest power of 2 that divides both u_0 and u_1. Define $g = G(v)$, and let $S = \{u \in [\sqrt{n}]^2 \mid G(u) = g\}$. We have that Mv is uniform over S: $\mathcal{M}$ is a group and S is the orbit of $(0, g)$.

Because S lies on a lattice of distance g and does not include the origin, there are at most $(2 \lfloor C/g \rfloor + 1)^2 - 1 \leq 8(C/g)^2$ elements in $S \cap [-C, C]^2$, and $(3/4)n/g^2$ total elements in S. Hence the probability is at most $(32/3)C^2/n$. □

We can then define the "hash function" $h_{M,b} : [\sqrt{n}]^2 \to [\sqrt{B}]^2$ given by $(h_{M,b}(u)) = \text{round}(\pi_{M,b}(u) \cdot \sqrt{n/B})$; i.e., round to the nearest multiple of $\sqrt{n/B}$ in each coordinate and scale down. We also define the "offset" $o_{M,b}(u) = \pi_{M,b}(u) - \sqrt{n/B} h_{M,b}(u)$. This lets us give results analogous to Claims 3.1 and 3.2 of [HIKP12a]:

- $\Pr[h_{M,b}(u) = h_{M,b}(v) < O(1/B)]$ for $u \neq v$. In order for $h(u) = h(v)$, we need that $\pi_{M,b}(u) - \pi_{M,b}(v) \in [-2\sqrt{n/B}, 2\sqrt{n/B}]^2$. But Lemma B.4 implies this probability is $O(1/B)$.

- $\Pr[o_{M,b}(u) \notin [-(1-\alpha)\sqrt{n/B}, (1-\alpha)\sqrt{n/B}]^2] < O(\alpha)$ for any $\alpha > 0$. Because of the offset b, $o_{M,b}(u)$ is uniform over $[-\sqrt{n/B}, \sqrt{n/B}]^2$. Hence the probability is $2\alpha - \alpha^2 + o(1)$ by a volume argument.

which are all we need of the hash function.

Filter Modifying the filter is pretty simple. Specifically,[HIKP12a] defined a filter $G \in \mathbb{R}^{\sqrt{n}}$ with support size $O(\sqrt{B} \log n)$ such that $\widehat{G}$ is essentially zero outsize $[-\sqrt{n/B}, \sqrt{n/B}]$ and is essentially 1 inside $[-(1-\alpha)\sqrt{n/B}, (1-\alpha)\sqrt{n/B}]$ for constant α. We compute the $\sqrt{B} \times \sqrt{B}$ 2-dimensional DFT of $x'_{i,j} = x_{i,j} G_i G_j$ to sum up the element in each bin. This takes $B \log^2 n$ samples and time rather than $B \log n$, which is the reason for the extra $\log n$ factor compared to the one dimensional case.

Location Location is easy to modify; we simply run it twice to find the row and column separately.

In summary, the aforementioned adaptations leads to a variant of the [HIKP12a] algorithm that works in two dimensions, with running time $O(k \log(n/k) \log^2 n)$, using $O(k \log(n/k) \log^2 n)$ samples.

Adding extra coefficient list $\widehat{z}$ The modification of the algorithm of [HIKP12a] (as well as its variant above) is straightforward. The algorithm performs a sequence of iterations, where each iteration involves hashing the frequencies of the signal into bins, followed by subtracting the already recovered coefficients from the bins. Since the algorithm recovers $\Theta(k)$ coefficients in the first iteration, the subtracted list is always of size $\Theta(k)$.

Given the extra coefficient list, the only modification to the algorithm is that the list of the subtracted coefficients needs to be appended with coefficients in $\widehat{z}$. Since this step does not affect the samples taken by the algorithm, the sample bound remains unchanged. To analyze the running time, let k' be the number of nonzero coefficients in $\widehat{z}$. Observe that the total time of the original algorithm spent on subtracting the coefficients from a list of size $\Theta(k)$ was $O(k \log(n/k) \log n)$, or $O(\log(n/k) \log n)$ per list coefficient. Since in our case the number of coefficients in the list is increased from $\Theta(k)$ to $k' + \Theta(k)$, the running time is increased by an additive factor of $O(k' \log(n/k) \log n)$. □