Combinatorial Sublinear-Time Fourier Algorithms[*]

M. A. Iwen[†]
Institute for Mathematics and its Applications (IMA)
University of Minnesota
iwen@ima.umn.edu

July 29, 2009

Abstract

We study the problem of estimating the best k term Fourier representation for a given frequency-sparse signal (i.e., vector) $\mathbf{A}$ of length $N \gg k$. More explicitly, we investigate how to deterministically identify k of the largest magnitude frequencies of $\hat{\mathbf{A}}$, and estimate their coefficients, in polynomial$(k, \log N)$ time. Randomized sublinear time algorithms which have a small (controllable) probability of failure for each processed signal exist for solving this problem [24, 25]. In this paper we develop the first known deterministic sublinear time sparse Fourier Transform algorithm which is guaranteed to produce accurate results. As an added bonus, a simple relaxation of our deterministic Fourier result leads to a new Monte Carlo Fourier algorithm with similar runtime/sampling bounds to the current best randomized Fourier method [25]. Finally, the Fourier algorithm we develop here implies a simpler optimized version of the deterministic compressed sensing method previously developed in [30].

1 Introduction

Suppose we are given a periodic function $f : [0, 2\pi] \to \mathbb{C}$ which is well approximated by a k-sparse trigonometric polynomial

$$\tilde{f}(x) = \sum_{j=1}^{k} C_j e^{\mathbb{i}\omega_j \cdot x}, \ \{\omega_1, \dots, \omega_k\} \subset \left[-\frac{N}{2}, \frac{N}{2}\right] \cap \mathbb{Z}, \tag{1}$$

where the smallest such N is much larger than k. We seek simple methods for quickly recovering a high-fidelity approximation to $\tilde{f}$ using as few evaluations of f as possible. In the course of developing four such methods, we explore the interplay between three generally conflicting goals: (*i*) deterministic recovery with guarantees, (*ii*) minimized recovery runtime, and (*iii*) minimized evaluations of f.

Compressed Sensing (CS) methods [6, 15, 5, 50, 46, 35] provide a robust framework for reducing the number of samples required to estimate a periodic function's Fourier transform (FT). Results on Gelfand widths establish these CS results as being near-optimal with respect to minimal sampling requirements (see [8], and [49, 16, 35]). For this reason CS methods have proven useful for reducing high sampling costs in applications such as analog-to-digital conversion [36, 34] and MR imaging [38, 39]. However, despite small sampling requirements, standard CS Fourier methods utilizing Basis Pursuit (BP) [15, 5, 14] and Orthogonal Matching Pursuit (OMP) [50, 46] have runtime requirements which are superlinear in the function's bandwidth N. We seek to develop methods for applications involving the interpolation of high-bandwidth functions where runtime is of primary importance (e.g., CS-based numerical methods for multiscale problems [13]). Hence, we focus on developing algorithms with runtime complexities that scale sublinearly (e.g., logarithmically) in the bandwidth of the input function.

A second body of work on compressed sensing includes methods which have achieved near-optimal reconstruction runtime bounds [24, 25, 11, 12, 45, 27, 29]. However, with the notable exception of [24, 25], these CS algorithms do

[*]Results herein supersede preliminary Fourier results in [30, 32].
[†]This research was supported in part by NSF DMS-0510203.

not permit sublinear sampling in the Fourier case. Despite their efficient reconstruction algorithms, their total Fourier measurement and reconstruction runtime costs are still superlinear in the signal size/bandwidth. In the Fourier case they generally require more operations than a standard FFT for all nontrivial sparsity levels while utilizing approximately the same number of signal samples.

To date, only randomized Fourier methods [24, 25] have been shown to outperform the FFT in terms of runtime on frequency-sparse broadband superpositions while utilizing only a fraction of the FFT's required samples [31]. However, they employ random sampling techniques and thus fail to output good approximate answers with non-zero probability. Furthermore, they are perhaps unnecessarily complicated to implement and optimize in practice. Related work includes earlier methods for the sublinear-time reconstruction of sparse trigonometric polynomials via random sampling [41, 7]. In turn, these methods can be traced back further to algorithms for learning sparse multivariate polynomials over fields of characteristic zero [20, 42].

Finally, our CS-recovery techniques in Section 5 are related to group testing methods [17]. In particular, our k-majority separating collection of sets construction is related to the number theoretic group testing construction utilized in [19]. This relationship to group testing, in combination with the Fourier transform's natural aliasing behavior, is essentially what allows our sublinear-time Fourier methods to be constructed. For more on group testing and signal recovery see [26].

1.1 Illustrative Examples

In this section we present some simple examples in order to illustrate the methods which will ultimately allow us to construct sublinear-time sparse Fourier algorithms. We begin by outlining a means for recovering functions consisting of one energetic frequency (e.g., a cosine function in the real case). We then outline how to extend these ideas to allow the recovery of more complicated trigonometric polynomials containing two energetic frequencies. Along the way we introduce terminology and useful concepts used in subsequent sections.

1.1.1 Singe Frequency Functions

Let $f : [0, 2\pi] \to \mathbb{C}$ be a non-identically zero function of the form

$$f(x) = C \cdot e^{\mathbb{i}\omega x}$$

consisting of a single unknown frequency $\omega \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \cap \mathbb{Z}$. Recovering f with a standard FFT would dictate the need for at least N equally spaced samples from $[0, 2\pi)$ to avoid aliasing effects (see [4]). Thus, we would have to compute the FFT of the N-length vector

$$\mathbf{A}(j) = f\left(\frac{2\pi j}{N}\right), \;\; j \in [0, N) \cap \mathbb{N}.$$

However, if we use aliasing to our advantage, we can correctly determine ω with significantly fewer f-samples as follows.

Let $\mathbf{A}_2$ be the 2-element array of equally spaced f-samples from $[0, 2\pi)$. Thus, we have

$$\mathbf{A}_2(0) = f(0) = C, \text{ and } \mathbf{A}_2(1) = f(\pi) = C \cdot (-1)^{\omega}.$$

Calculating the discrete Fourier transform of $\mathbf{A}_2$, denoted $\widehat{\mathbf{A}_2}$, we get that

$$\widehat{\mathbf{A}_2}(0) = C \cdot \pi \cdot (1 + (-1)^{\omega}), \text{ and } \widehat{\mathbf{A}_2}(1) = C \cdot \pi \cdot \left(1 + (-1)^{\omega+1}\right).$$

Note that since ω is an integer, exactly one element of $\widehat{\mathbf{A}_2}$ will be non-zero. If $\widehat{\mathbf{A}_2}(0) \neq 0$ then we know that $\omega \equiv 0$ modulo 2. On the other hand, $\widehat{\mathbf{A}_2}(1) \neq 0$ implies that $\omega \equiv 1$ modulo 2. We say that

$$\hat{f}(\omega) = \int_0^{2\pi} f(x)\, e^{-\mathbb{i}\omega x}\, dx = 2\pi \cdot C$$

has been *aliased* to $\widehat{\mathbf{A}_2}(\omega \bmod 2)$. Note that by computing $\widehat{\mathbf{A}_2}$ we have effectively hashed the Fourier transform of f into a 2 element array.

More generally, we may replace 2 by any natural number in the paragraph above and obtain the same aliasing-based hashing behavior. Thus, we may compute several potentially aliased Fast Fourier Transforms in parallel to discover ω modulo $3, 5, \ldots,$ the $O(\log N)^{\text{th}}$ prime. Once we have collected these moduli we can reconstruct ω via the famous Chinese Remainder Theorem (see [47]).

Theorem 1. CHINESE REMAINDER THEOREM (CRT): *Any integer x is uniquely specified mod N by its remainders modulo m relatively prime integers $p_1, \ldots, p_m$ as long as $\prod_{l=1}^{m} p_l \geq N$.*

To finish our example, suppose that $N = 10^6$ and that we have used three FFT's with 100, 101, and 103 samples to determine that $\omega \equiv 34 \bmod 100$, $\omega \equiv 3 \bmod 101$, and $\omega \equiv 1 \bmod 103$, respectively. Using that $\omega \equiv 1 \bmod 103$ we can see that $\omega = 103 \cdot a + 1$ for some integer a. Using this new expression for ω in our second modulus we get

$$(103 \cdot a + 1) \equiv 3 \bmod 101 \Rightarrow a \equiv 1 \bmod 101.$$

Therefore, $a = 101 \cdot b + 1$ for some integer b. Substituting for a we get that $\omega = 10403 \cdot b + 104$. By similar work we can see that $b \equiv 10 \bmod 100$ after considering ω modulo 100. Hence, $\omega = 104,134$ by the CRT. As an added bonus we note that our three FFTs will have also provided us with three different estimates of ω's coefficient C.

The end result is that we have used significantly less than N samples to determine ω. Using the CRT we required only $100 + 101 + 103 = 304$ samples from f to determine ω since $100 \cdot 101 \cdot 103 > 10^6$. In contrast, a million f-samples would be gathered during recovery with a standard FFT. Besides needing significantly less samples than the FFT, this CRT-based single frequency method dramatically reduces the required computational effort. Moreover, it is deterministic. There is no chance of failure. Of course, a single frequency signal is incredibly simple. Signals involving more than 1 non-zero frequency are much more difficult to handle since frequency moduli may begin to collide modulo various numbers. Dealing with the potential difficulties caused by such frequency collisions in a deterministic way comprises the majority of this paper.

1.1.2 Two Frequencies

Suppose, for simplicity, that we now wish to recover a function, $f : [0, 2\pi] \to \mathbb{C}$, consisting of two unknown frequencies $\omega_1, \omega_2 \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \cap \mathbb{Z}$. Hence, we have

$$f(x) = C_1 \cdot e^{\mathbb{i}\omega_1 x} + C_2 \cdot e^{\mathbb{i}\omega_2 x}.$$

Furthermore, assume for the moment that we have discovered a natural number n such that

$$\omega_1 \bmod n \neq \omega_2 \bmod n.$$

In this case we will say that *n separates* ω_1 and ω_2. Our current goal is to demonstrate how we may obtain information about the identities of both ω_1 and ω_2 using n in combination with several small discrete Fourier transforms.

Fix $c \in \mathbb{N}$ and let $h_1 = \omega_1 \bmod n$, $h_2 = \omega_2 \bmod n$. We will refer to both h_1 and h_2 as *residues*, or *remainders*, modulo n. If we form an array of $c \cdot n$ equally spaced samples

$$\mathbf{A}_{c \cdot n}(j) = f\left(\frac{2\pi j}{cn}\right), \ \ j \in [0, n) \cap \mathbb{Z},$$

we can see from our discussion of aliasing in Section 1.1.1 that $\widehat{\mathbf{A}_{c\cdot n}}$ will have 0 in every entry except for the two entries

$$\widehat{\mathbf{A}_{c\cdot n}}(\omega_1 \bmod cn) = C_1 \neq 0, \text{ and } \widehat{\mathbf{A}_{c\cdot n}}(\omega_2 \bmod cn) = C_2 \neq 0.$$

Thus, we can determine both h_1 and h_2 by considering

$$\left|\widehat{\mathbf{A}_n}(0)\right|, \left|\widehat{\mathbf{A}_n}(1)\right|, \ldots, \left|\widehat{\mathbf{A}_n}(n-1)\right|.$$

The two largest entries will be the residues h_1 and h_2.

We would like to repeat the procedure described in the last paragraph for several different values of c, thereby learning both ω_1 and ω_2 modulo several different cn values. We can then imagine being able to reconstruct both ω_1 and ω_2 using the CRT as per Section 1.1.1. However, before this approach can work, we must deal with two difficulties: First, the procedure as described above will not yield residues of ω_1 and ω_2 modulo relatively prime integers since n divides cn for all $c \in \mathbb{Z}$. Second, in the worst case (e.g., if $C_1 = C_2$), we will not be able to determine which residues modulo each cn-value correspond to ω_1 versus ω_2. If we mismatch residues from ω_1 and ω_2 during our CRT based reconstruction we could obtain a potentially huge number of bogus frequencies. The following procedure, inspired by work from [45, 11, 12], circumvents both of these difficulties.

Suppose we have calculated both $\widehat{\mathbf{A}_n}$ and $\widehat{\mathbf{A}_{2\cdot n}}$. Focusing on ω_1, we know that

$$\widehat{\mathbf{A}_{2\cdot n}}(\omega_1 \bmod 2n) = C_1 = \widehat{\mathbf{A}_n}(h_1).$$

Since $\omega_1 = h_1 + a \cdot n$ for some $a \in \mathbb{Z}$, we can see that either

$$\omega_1 \bmod 2n = h_1, \text{ or } \omega_1 \bmod 2n = h_1 + n$$

must be true depending on whether a is even or odd, respectively. Therefore, we may determine $\omega_1 \bmod 2n$ by

$$\omega_1 \bmod 2n = \begin{cases} h_1 & \text{if } \left|\widehat{\mathbf{A}_n}(h_1) - \widehat{\mathbf{A}_{2\cdot n}}(h_1)\right| < \left|\widehat{\mathbf{A}_n}(h_1) - \widehat{\mathbf{A}_{2\cdot n}}(h_1+n)\right| \\ h_1 + n & \text{otherwise} \end{cases} . \tag{2}$$

Finally, once we have calculated $\omega_1 \bmod 2n$ we may calculate $\omega_1 \bmod 2$ by

$$\omega_1 \bmod 2 = (\omega_1 \bmod 2n) \bmod 2.$$

Furthermore, if 2 is relatively prime to n, we can use both these ω_1 residues together in the CRT. Using several $\widehat{\mathbf{A}_{c\cdot n}}$ in this fashion we can discover ω_1 modulo $c = 2, 3, 5, \ldots,$ the $O(\log N)^{\text{th}}$ prime. Once we have collected these moduli we can then reconstruct ω_1 as discussed in Section 1.1.1. Of course, ω_2 can be reconstructed in a similar manner.

Note that we used the assumption that n separates ω_1 and ω_2 throughout this discussion. If, for example, both $h_1 = h_2$ and $C_1 = C_2$ are true, then Equation 2 is not guaranteed to hold. In this case we might incorrectly calculate $\omega_1 \bmod cn$ for one of our $c = 2, 3, \ldots$ values, resulting in the incorrect recovery of ω_1 and/or ω_2. In the next section we discuss strategies for dealing with the fact that n is not generally known to possess the desired separation properties.

1.1.3 Separating Two Frequencies

In Section 1.1.2 we assumed we knew a small natural number n that separated the two unknown frequencies $\omega_1, \omega_2 \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \cap \mathbb{Z}$. Given such an n, we outlined how to reconstruct both ω_1 and ω_2 using a few aliased discrete Fourier transforms of small size. In this section we will utilize an example to quickly sketch how we can recover ω_1 and ω_2 even if we do not have any knowledge concerning separating n-values. The techniques below were motivated by similar number theoretic group testing strategies employed in [44, 19].

Suppose that $N = 10^6$, and let $n_1, n_2, n_3, n_4,$ and n_5 be the five relatively prime values 100, 101, 103, 107, and 109, respectively. Clearly the product of any three (or more) of these five n-values is larger than N. As a consequence, if $\omega_1 \equiv \omega_2 \bmod n_j$ for more than two unique j values in $\{1, 2, 3, 4, 5\}$ then $\omega_1 = \omega_2$ by the Chinese Remainder Theorem. To see why this is true, note that $\omega_1 = \omega_1 \bmod N$, $\omega_2 = \omega_2 \bmod N$, and any three of our n_j-values are relatively prime with product larger than N. Therefore, the CRT dictates that any two distinct integers ω_1 and ω_2 can *collide* (i.e., have the same remainders) modulo at most two of our relatively prime n_j-values. If ω_1 and ω_2 collide more than twice they must be equal.

We are now in the happy position of knowing that the majority of our five n_j-values must separate ω_1 and ω_2. However, we do not know which 3 or more of the 5 n_j-values are good separating values. In this simple two frequency case we could find a separating value by, for example, choosing one of the five values,

$$\tilde{n}_j \in \{100,\ 101,\ 103,\ 107,\ 109\},$$

uniformly at random and then seeing if its associated discrete Fourier transform, $\widehat{\mathbf{A}_{\tilde{n}_j}}$, contains two nonzero entries. If $\widehat{\mathbf{A}_{\tilde{n}_j}}$ contains two nonzero values, it would mean that $\tilde{n}_j$ separates ω_1 and ω_2. Methods from Section 1.1.2 could then be used to recover both frequencies. Furthermore, we expect to have to test only two randomly chosen n_j-values before finding one that separates ω_1 and ω_2.

A related deterministic approach simply temporarily assumes that all five n_j-values separate ω_1 and ω_2. The recovery methods from Section 1.1.2 are then applied once for each of the five n_j-values (i.e., with $n = 100$, 101, 103, 107, and 109). The fact that a couple n_j-values may not actually separate the two frequencies can be accounted for later by inspecting the recovered answers. Each time the Section 1.1.2 recovery methods are applied with a different n_j-value, at most two unique frequency answers are returned, for a total of at most 10 answers (with multiplicity). Within these at most 10 answers, ω_1 and ω_2 must both appear at least three times each. This is because at least 3 of our 5 n_j-values must separate ω_1 and ω_2, and whenever separation occurs our Section 1.1.2 methods are guaranteed to return the correct answers. Hence, at least 6 of our at most 10 answers are guaranteed to be correct. The at most 4 remaining incorrect answers will contain at most 4 incorrectly recovered frequencies, each of which can appear at most twice. The end result is that (*i*) both ω_1 and ω_2 will appear in our list of at most 10 answers at least 3 times each, and (*ii*) no incorrect frequency can appear in our list of answers more than twice. Hence, if we return all answers which appear 3 or more times we will get exactly ω_1 and ω_2.

To conclude, recall from Section 1.1.2 that whenever an n_j-value separates ω_1 and ω_2, the aliased discrete Fourier transform $\widehat{\mathbf{A}_{n_j}}$ will contain two nonzero entries: one equal to $\hat{f}(\omega_1) = 2\pi \cdot C_1$ and another equal to $\hat{f}(\omega_2) = 2\pi \cdot C_2$. Thus, both $\hat{f}(\omega_1)$ and $\hat{f}(\omega_2)$ will be calculated correctly in at least three of $\widehat{\mathbf{A}_{100}}$, $\widehat{\mathbf{A}_{101}}$, $\widehat{\mathbf{A}_{103}}$, $\widehat{\mathbf{A}_{107}}$, and $\widehat{\mathbf{A}_{109}}$. Given that we have identified ω_1 and ω_2, at this point it is easy to find one of the at least 3 separating n_j-values by calculating

$$\omega_1 \bmod 100, \ \ldots, \ \omega_1 \bmod 109,$$

and

$$\omega_2 \bmod 100, \ \ldots, \ \omega_2 \bmod 109.$$

Once a separating n_j-value, n, has been found, we can calculate $\hat{f}(\omega_1)$ and $\hat{f}(\omega_2)$ by

$$\hat{f}(\omega_1) = \widehat{\mathbf{A}_n}(\omega_1 \bmod n), \text{ and } \hat{f}(\omega_2) = \widehat{\mathbf{A}_n}(\omega_2 \bmod n).$$

We are now equipped with methods for recovering any 2-frequency superposition. More importantly, we will see in subsequent sections that the ideas we have employed here extend naturally to allow the approximate recovery of more complicated functions.

1.2 Results and Related Work

In this paper we construct the first known deterministic sublinear-time sparse Fourier algorithm guaranteed to produce accurate approximations of frequency-sparse signals. In order to produce our new Fourier algorithm we introduce a combinatorial object called a k-majority separating collection of sets which can be constructed using number theoretic methods. The number theoretic nature of our construction allows the sublinear-time computation of Fourier measurements via aliasing. Finally, a simple relaxation of our deterministic Fourier method provides a new randomized Fourier algorithm with near-optimal sampling/runtime requirements for k-sparse signals (i.e., signals with at most k nonzero frequencies).

Table 1 compares the Fourier algorithms developed in this paper to existing sparse Fourier methods. All the methods listed are robust with respect to noise (i.e., are robust trigonometric interpolation methods). The runtime and sampling requirements are for recovering exact k-sparse trigonometric polynomials (see Equation 1). The second column indicates whether the result recovers (up to machine precision) the input signal with high probability (w.h.p.), or deterministically (D) with guarantees. "With high probability" indicates a nonuniform $\frac{1}{N^{\Theta(1)}}$ failure probability per signal. In some cases, for simplicity, a factor of "$\log(k)$" or "$\log(N/k)$" was weakened to "$\log(N)$".

Looking at Table 1 we can see that our randomized recovery results in Section 4 require an additional $\log(N)$ factor in terms of sampling complexity over both the Linear Programming (LP) and Orthogonal Matching Pursuit (OMP) reconstruction methods. However, the Section 4 algorithms are simpler to implement and have lower guaranteed runtime

Fourier Algorithm	w.h.p./D	Runtime	Function Samples
LP [6] or OMP [50]	w.h.p.	$N^{O(1)}$	$O(k \cdot \log(N))$ [5, 35]
Section 4	w.h.p.	$O(N \cdot \log^3(N))$	$O(k \cdot \log^2(N))$
Section 4	D	$O(N \cdot k \cdot \log^2(N))$	$O(k^2 \cdot \log^2 N)$
Sparse Fourier [25]	w.h.p.	$O(k \cdot \log^{O(1)}(N))$	$O(k \cdot \log^{O(1)}(N))$
Section 5	w.h.p.	$O(k \cdot \log^5(N))$	$O(k \cdot \log^4(N))$
Section 5	D	$O(k^2 \cdot \log^4(N))$	$O(k^2 \cdot \log^4(N))$

Table 1: Sparse Fourier Algorithms

complexity. The Section 4 algorithms are also capable of exactly reconstructing k-sparse signals in an exact arithmetic setting. More interestingly, we note that our Monte Carlo result in Section 5 has similar runtime requirements to the sparse Fourier method developed in [25].

1.2.1 Sublinear-time Methods: Approximation Guarantees

Of the aforementioned compressed sensing methods, our deterministic Fourier results are most closely related to the work of Cormode and Muthukrishnan (CM) [12, 45, 11]. In effect, the deterministic sublinear-time Fourier methods in Section 5 are improved versions of CM's deterministic compressed sensing results which have been specifically adapted to the Fourier CS setting. Thus, the results in Section 5 exhibit similar approximation guarantees to the results of CM despite the fact that CM's methods do not themselves directly yield sparse Fourier algorithms.

Fix $p, c \in \mathbb{R}^+$. We will refer to any periodic function whose b^{th}-largest magnitude Fourier coefficient is less than $c \cdot b^{-p}$ for all $b \in \mathbb{N}$ as (c, p)-*compressible*. For given c, p values, both CM and this paper provide sparse approximation guarantees in terms of the min-max error over all (c, p)-compressible functions. In essence, the Fourier Transform (FT) of any input (c, p)-compressible function is guaranteed to be approximated with error at most as large as the error incurred by approximating the worst possible (c, p)-compressible function's FT by its best possible k-term Fourier representation (see Section 2 for details). These types of compressible approximation guarantees have a long history. See Chapter 9 of [40] for more about approximating signals that are (c, p)-compressible in various bases (e.g., Fourier, wavelet, etc).

Given a (c, p)-compressible function with $p > 2$, CM implicitly provide a two stage algorithm which produces an accurate Fourier approximation consisting of k-frequencies $\in \left[-\frac{N}{2}, \frac{N}{2}\right] \cap \mathbb{Z}$. In the first stage an array containing $O\left(k^{\frac{4p}{p-2}} \log^4 N\right)$ entries is calculated using $\Omega(N)$-time/signal samples. In the second stage, a $O\left(k^{\frac{6p}{p-2}} \log^6 N\right)$ time algorithm uses the array from the first stage to generate the k-term Fourier approximation. In contrast, our deterministic methods developed in Section 5 can recover a k-term Fourier approximation with the same error guarantees in $O\left(k^{\frac{2p}{p-1}} \cdot \log^6 N\right)$ time. This runtime is both sublinear in N and faster than the second stage of CM's algorithm for all $p > 2$. Furthermore, the number of signal samples used is $O\left(k^{\frac{2p}{p-1}} \cdot \log^5 N\right)$, also sublinear in N.

Given any periodic function f, we can best approximate its Fourier transform using k of its largest magnitude Fourier coefficients. Let $\tilde{f}$ denote the associated optimal k-sparse trigonometric polynomial approximation to f (e.g., see Equation 1). The randomized sparse Fourier transform algorithms developed in [24, 25] can, for example, produce a k-term Fourier approximation, $\hat{\mathbf{R}}$, for each input function f which has

$$\left\|\hat{f} - \hat{\mathbf{R}}\right\|_2 \leq \sqrt{2} \cdot \left\|\hat{f} - \hat{\tilde{f}}\right\|_2$$

with high probability. Note that this approximation result is quite strong. It states that the output, $\hat{\mathbf{R}}$, is highly likely to be near-optimal with respect to *each input function f*. Unfortunately, it is impossible for a deterministic sublinear-time Fourier algorithm to obtain this type of strong l^2-approximation guarantee (see Section 5 of [8]). Thus, we focus on the weaker min-max approximation guarantees proven for the sublinear-time algorithms developed in Section 5.

The remainder of this paper is organized as follows: In Section 2 we introduce relevant definitions, terminology, and background. Then, in Section 3 we define k-majority selective collections of sets and present number theoretic constructions. In Section 4 we use k-majority selective collections of sets to develop simple superlinear-time Fourier algorithms. We also present analysis of these algorithms' runtime and sampling requirements. In Section 5 we modify the results from Section 4 in order to produce sublinear-time Fourier algorithms. The discrete versions of these algorithms are briefly presented in Section 6. Finally, a preliminary empirical evaluation of the deterministic algorithms is included in Section 7. Section 8 contains a short conclusion.

2 Preliminaries

Throughout the remainder of this paper we will be interested in complex valued functions $f : [0, 2\pi] \mapsto \mathbb{C}$ and signals (or arrays) of length N containing f values at various $x \in [0, 2\pi]$. We shall denote such signals by $\mathbf{A}$, where $\mathbf{A}(j) \in \mathbb{C}$ is the signal's j^{th} complex value for all $j \in [0, N) \cap \mathbb{Z}$. Hereafter we will refer to the process of either calculating, measuring, or retrieving the f value associated with any $\mathbf{A}(j) \in \mathbb{C}$ from machine memory as *sampling* from f and/or $\mathbf{A}$. Finally, throughout the remainder of the paper $[0, N)$ will always denote $[0, N) \cap \mathbb{Z}$. Likewise, $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ will always stand for $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \cap \mathbb{Z}$.

Given a signal $\mathbf{A}$, we define its discrete l^q-norm to be

$$||\mathbf{A}||_q = \left(\sum_{j=0}^{N-1} |\mathbf{A}(j)|^q \right)^{\frac{1}{q}}. \tag{3}$$

More specifically, we will refer to $||\mathbf{A}||_2^2$ as $\mathbf{A}$'s *energy*. We will say that $\mathbf{A} \in l^q$ if $||\mathbf{A}||_q^q$ converges (i.e., we allow $N = \infty$). Finally, j and ω will always denote integers.

2.1 Compressed Sensing and Compressibility

Given a signal $\mathbf{A}$, let Ψ be an invertible $N \times N$ matrix under which $\mathbf{A}$ is sparse (i.e., only $k \ll N$ entries of $\Psi \cdot \mathbf{A}$ are significant/large in magnitude). Compressed sensing (CS) methods [24, 25, 11, 12, 45, 27, 29, 30] deal with generating a $K \times N$ measurement matrix, $\mathcal{M}$, with the smallest number of rows possible (i.e., K minimized) so that the k significant entries of $\Psi \cdot \mathbf{A}$ can be well approximated by a recovery algorithm using only the K-element vector result of

$$(\mathcal{M} \cdot \Psi) \cdot \mathbf{A} \tag{4}$$

as input. Thus, CS methods attempt to approximate $\Psi \cdot \mathbf{A}$ using only a compressed set of measurements. For the purposes of this paper our compressed set of measurements will take the form of a compressed (i.e., sublinear) set of function samples.

A fast CS recovery algorithm will output k tuples from $[0, N) \times \mathbb{C}$. We will refer to any such set of $k < N$ tuples

$$\left\{ (\tilde{\omega}_l, C_l) \in [0, N) \times \mathbb{C} \,\middle|\, l \in [1, k] \right\}$$

as a **sparse Ψ representation** and denote it with a superscript 's'. Note that if we are given a sparse Ψ representation, $\mathbf{R}^{\mathrm{s}}_{\Psi}$, we may consider $\mathbf{R}^{\mathrm{s}}_{\Psi}$ to be a length-N signal. We simply view $\mathbf{R}^{\mathrm{s}}_{\Psi}$ as the N length signal

$$\mathbf{R}_{\Psi}(j) = \begin{cases} C_j & \text{if } (j, C_j) \in \mathbf{R}^{\mathrm{s}}_{\Psi} \\ 0 & \text{otherwise} \end{cases}$$

for all $j \in [0, N)$. Using this idea, we may reconstruct $\mathbf{R}$ in any desired basis using $\mathbf{R}^{\mathrm{s}}_{\Psi}$.

A k term/tuple sparse Ψ representation is k-optimal for a signal $\mathbf{A}$ if it contains k of the largest magnitude entries of $\Psi \cdot \mathbf{A}$. More precisely, we'll say that a sparse Ψ representation $\mathbf{R}^{\mathrm{s}}_{\Psi}$ is **k-optimal** for $\mathbf{A}$ if there exists a valid ordering of $\Psi \cdot \mathbf{A}$ by magnitude

$$\left| (\Psi \cdot \mathbf{A})(\omega_1) \right| \geq \left| (\Psi \cdot \mathbf{A})(\omega_2) \right| \geq \cdots \geq \left| (\Psi \cdot \mathbf{A})(\omega_j) \right| \geq \cdots \geq \left| (\Psi \cdot \mathbf{A})(\omega_N) \right| \tag{5}$$

so that $\left\{ (\omega_l, (\Psi \cdot \mathbf{A})(\omega_l)) \;\middle|\; l \in [1,k] \right\} = \mathbf{R}^{\mathrm{s}}_{\Psi}$. Note that a signal $\mathbf{A}$ may have several k-optimal Ψ representations if its $\Psi \cdot \mathbf{A}$ entry magnitudes are non-unique. For example, there are two 1-optimal sparse Fourier representations for the signal

$$\mathbf{A}(j) = 2e^{\frac{-2\pi \mathbb{i} j}{N}} + 2e^{\frac{2\pi \mathbb{i} j}{N}},\ N > 2.$$

However, all k-optimal Ψ representations, $\mathbf{R}^{\mathrm{s}}_{\mathrm{opt}\ \Psi}$, for any signal $\mathbf{A}$ will always have both the same unique $\|\mathbf{R}_{\mathrm{opt}\ \Psi}\|_2$ and $\|(\Psi \cdot \mathbf{A}) - \mathbf{R}_{\mathrm{opt}\ \Psi}\|_2$ values.

We conclude this subsection with two definitions: Let ω_b be a b^{th} largest magnitude entry of $\Psi \cdot \mathbf{A}$ as per Equation 5. We will say that a signal $\Psi \cdot \mathbf{A}$ is **(algebraically)** (c,p)**-compressible** for some $c, p \in \mathbb{R}^+$ if $|(\Psi \cdot \mathbf{A})(\omega_b)| \leq c \cdot b^{-p}$ for all $b \in [1,N]$. If $\mathbf{R}^{\mathrm{s}}_{\mathrm{opt}\ \Psi}$ is a k-optimal Ψ representation we can see that

$$\|(\Psi \cdot \mathbf{A}) - \mathbf{R}_{\mathrm{opt}\ \Psi}\|_2^2 = \sum_{b=k+1}^{N} \left|(\Psi \cdot \mathbf{A})(\omega_b)\right|^2 \leq c^2 \cdot \int_k^\infty b^{-2p} db = \frac{c^2}{2p-1} \cdot k^{1-2p} \tag{6}$$

as long as $p > 1/2$. For any (c,p)-compressible signal class (i.e., for any choice of $p > 1/2$ and $c \in \mathbb{R}^+$) we will refer to the related optimal $O(k^{1-2p})$-size worst case error value (i.e., see Equation 6 above) as $\|C_k^{\mathrm{opt}}\|_2^2$. Similarly, we define an **exponentially compressible** (or **exponentially decaying**) signal for fixed $c, \alpha \in \mathbb{R}^+$ to be one for which $\left|(\Psi \cdot \mathbf{A})(\omega_b)\right| \leq c \cdot e^{-\alpha b}$ for all $b \in [1,N]$. The optimal worst case error is then

$$\|C_k^{\mathrm{opt}}\|_2^2 \leq c^2 \cdot \int_k^\infty e^{-2\alpha b} db = \frac{c^2}{2\alpha} \cdot e^{-2\alpha k}. \tag{7}$$

2.2 The Fourier Case

For the remainder of this paper we will be interested the special CS case where Ψ is the $N \times N$ Discrete Fourier Transform (DFT) matrix. In this case we have

$$\Psi_{i,j} = \frac{2\pi}{N} \cdot e^{\frac{2\pi \mathbb{i} \cdot i \cdot j}{N}}. \tag{8}$$

Thus, the DFT of $\mathbf{A}$, denoted $\widehat{\mathbf{A}}$, is another signal of length N defined as follows:

$$\widehat{\mathbf{A}}(\omega) = \frac{2\pi}{N} \cdot \sum_{j=0}^{N-1} e^{\frac{-2\pi \mathbb{i} \omega j}{N}} \mathbf{A}(j), \quad \forall \omega \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]. \tag{9}$$

We will refer to any index, ω, of $\widehat{\mathbf{A}}$ as a *frequency*. Furthermore, we will refer to $\widehat{\mathbf{A}}(\omega)$ as frequency ω's *coefficient* for each $\omega \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$.

We may recover $\mathbf{A}$ from its DFT via the Inverse Discrete Fourier Transform (IDFT) as follows:

$$\mathbf{A}(j) = \widehat{\widehat{\mathbf{A}}}^{-1}(j) = \frac{1}{2\pi} \cdot \sum_{\omega = 1 - \lceil \frac{N}{2} \rceil}^{\lfloor \frac{N}{2} \rfloor} e^{\frac{2\pi \mathbb{i} \omega j}{N}} \widehat{\mathbf{A}}(\omega), \quad \forall j \in [0, N). \tag{10}$$

Parseval's identity tells us that $\|\widehat{\mathbf{A}}\|_2 = \sqrt{\frac{2\pi}{N}} \cdot \|\mathbf{A}\|_2$ for any signal. Note that any non-zero coefficient frequency will contribute to $\widehat{\mathbf{A}}$'s energy. Hence, we will also refer to $|\widehat{\mathbf{A}}(\omega)|^2$ as frequency ω's *energy*. If $|\widehat{\mathbf{A}}(\omega)|$ is relatively large, we will say that ω is *energetic*.

Fix accuracy parameter δ to be small (e.g., $\delta = 0.1$). Given an input signal, $\mathbf{A}$, with a compressible Fourier transform, our deterministic Fourier algorithm will identify k of the most energetic frequencies from $\widehat{\mathbf{A}}$ and approximate their coefficients to produce a sparse Fourier representation $\hat{\mathbf{R}}^{\mathrm{s}}$ with

$$\|\widehat{\mathbf{A}} - \hat{\mathbf{R}}\|_2^2 \leq \|\widehat{\mathbf{A}} - \hat{\mathbf{R}}_{\mathrm{opt}}\|_2^2 + \delta \|C_k^{\mathrm{opt}}\|_2^2. \tag{11}$$

It should be noted that the Fourier reconstruction algorithms below all extend naturally to the general compressed sensing case presented in Section 2.1 above via work analogous to that presented in [30].

3 Combinatorial Constructions

The following combinatorial structures are motivated by k-strongly separating sets [28, 11]. Their properties directly motivate our Fourier reconstruction procedures in Sections 4 and 5.

Definition 1. *A collection, $\mathcal{S}$, of subsets of $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ is called k-**majority selective** if both of the following are true: (i) $\cup_{S \in \mathcal{S}} S = \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, and (ii) for all $X \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ with $|X| \leq k$ and all $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, more than half of the subsets $S \in \mathcal{S}$ containing n are such that $S \cap X = \{n\} \cap X$ (i.e., every $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ occurs separated from all (other) members of X in more than half of the $\mathcal{S}$ elements containing n).*

Definition 2. *Fix an arbitrary $X \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ with $|X| \leq k$. A random collection of subsets of $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, $\mathcal{S}$, assembled independently of X is called (k, σ)-**majority selective** if both of the following are true with probability at least σ: (i) $\cup_{S \in \mathcal{S}} S = \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, and (ii) for all $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, more than half of the subsets $S \in \mathcal{S}$ containing n have the property that $S \cap X = \{n\} \cap X$ (i.e., with probability $\geq \sigma$ every $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ occurs separated from all (other) members of X in more than half of the $\mathcal{S}$ elements containing n).*

The existence of such sets is easy to see. For example, the collection of subsets

$$\mathcal{S} = \left\{ \{n\} \middle| n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \right\}$$

consisting of all the singleton subsets of $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ is k-majority selective for all $k \leq N$. Generally, however, we are interested in creating k-majority selective collections which contain as few subsets as possible (i.e., much fewer than N subsets). We next give a construction for a k-majority selective collection of subsets for any $k, N \in \mathbb{N}$ with $k \leq N$. Our construction is motivated by the prime groupings techniques first employed in [44]. We begin as follows:

Define $p_0 = 1$ and let p_l be the l^{th} prime natural number. Thus, we have

$$p_0 = 1, p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, \ldots$$

Choose $q, K \in \mathbb{N}$ (to be specified later). We are now ready to build a collection of subsets, $\mathcal{S}$. We begin by letting $S_{j,h}$ for all $0 \leq j \leq K$ and $0 \leq h \leq p_{q+j} - 1$ be

$$S_{j,h} = \left\{ n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \middle| n \equiv h \bmod p_{q+j} \right\}. \tag{12}$$

Next, we progressively define S_j to be all integer residues mod p_{q+j}, i.e.,

$$S_j = \{S_{j,h} \mid h \in [0, p_{q+j})\}, \tag{13}$$

and conclude by setting $\mathcal{S}$ equal to the union of all K such p_{q+j} residue groups:

$$\mathcal{S} = \cup_{j=0}^{K} S_j. \tag{14}$$

Hereafter $\mathcal{S}_j$*-primes* will refer to the set of $K + 1$ primes, $\{p_q, \ldots, p_{q+K}\}$, used to construct $\mathcal{S}$. We are now ready to prove that $\mathcal{S}$ is indeed k-majority selective if K is chosen appropriately.

Lemma 1. *Fix k. If we set $K \geq 2k \left\lfloor \log_{p_q} N \right\rfloor$, then $\mathcal{S}$ as constructed above will be a k-majority selective collection of sets.*

Proof:

Let $X \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ be such that $|X| \leq k$. Furthermore, choose $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ and let $x \in X$ be such that $x \neq n$. By the Chinese Remainder Theorem we know that x and n may only collide modulo at most $\lfloor \log_{p_q} N \rfloor$ of the $K + 1$

primes $p_{q+K} \geq \cdots \geq p_q$. This is because the product of any $\lfloor \log_{p_q} N \rfloor + 1$ $\mathcal{S}_j$-primes will be larger than N. Hence, there can be at most $k\lfloor \log_{p_q} N \rfloor$ $\mathcal{S}_j$-primes which fail to separate n from every element of $X - \{n\}$. We can now see that n will be isolated from all the (other) elements of X modulo at least $K + 1 - k\lfloor \log_{p_q} N \rfloor \geq k\lfloor \log_{p_q} N \rfloor + 1 > \frac{K+1}{2}$ $\mathcal{S}_j$-primes. Furthermore, n will appear in at most $K + 1$ subsets of $\mathcal{S}$. This leads us to the conclusion that $\mathcal{S}$ is indeed k-majority selective. □

Note that at least $\Omega(k)$ coprime integers are required in order to create a k-majority separating collection of subsets in this fashion. Given any $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$, the Chinese Remainder Theorem can be used to create a k element subset X with the property that each element of X collides with n in any desired $\Omega(1)$ $\mathcal{S}_j$-coprime numbers $\leq \frac{N}{2}$. Thus, it is not possible to significantly decrease the number of relatively prime values required to construct k-majority separating collections using these arguments.

The number of coprime integers required to construct each k-majority separating collection is directly related to the number of signal samples required by our subsequent Fourier algorithms. Given that we depend on the number theoretic nature of our constructions in order to take advantage of aliasing phenomena, it is unclear how to reduce the sampling complexity for our deterministic Fourier methods below. However, this does not stop us from appealing to randomized number theoretic constructions in order to decrease the number of required coprime values (and, therefore, samples). We next present a construction for (k, σ)-majority selective collections which motivates our subsequent Monte Carlo Fourier algorithms.

Lemma 2. *Suppose N is an integer grater than 3. Fix q, k, and an arbitrary $X \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ with $|X| \leq k$. We may form a (k, σ)-majority selective collection of subsets, $\mathcal{S}$, as follows: Set $K \geq 7k\left\lfloor \log_{p_q} N \right\rfloor$ and create a multiset $J \subset [0, K] \cap \mathbb{N}$ by independently choosing $21 \cdot \ln \frac{N}{1-\sigma}$ elements from $[0, K] \cap \mathbb{N}$ uniformly at random with replacement. Set $\mathcal{S} = \cup_{j \in J} S_j$ (see Equation 13).*

Proof:

Fix $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$. A prime chosen uniformly at random from $\{p_q, \ldots, p_{q+K}\}$ will separate n from all (other) elements of X with probability at least $\frac{6}{7}$. This is because at most $k\lfloor \log_{p_q} N \rfloor$ of the $K \geq 7k\lfloor \log_{p_q} N \rfloor$ $\mathcal{S}_j$-primes we select from can fail to separate n from every element of $X - \{n\}$ (see the proof of Lemma 1). Now consider the $|J|$ independent Poisson trails,

$$Y^n_1, \ldots, Y^n_m, \ldots, Y^n_{|J|},$$

related to the $|J|$ randomly selected $\mathcal{S}_j$-primes by

$$Y^n_m = \begin{cases} 1 & \text{if the } m^{\text{th}} \text{ selected } \mathcal{S}_j\text{-prime separates } n \text{ from } X - \{n\} \\ 0 & \text{otherwise} \end{cases}.$$

From above, the probability that each Y^n_m is 1 is a least $\frac{6}{7}$. Thus, $\mu = \mathbb{E}\left[\sum_{m=1}^{|J|} Y^n_m\right] \geq \frac{6 \cdot |J|}{7}$.

Using the Chernoff bound (see [43]) we get that the probability of

$$\sum_{m=1}^{|J|} Y^n_m < \frac{4}{7} \cdot |J|$$

is less than $e^{-\frac{\mu}{18}} \leq e^{-\frac{|J|}{21}} \leq \frac{1-\sigma}{N}$. Since $|J| > 21$, we can see that $\sum_{m=1}^{|J|} Y^n_m$ will be less than $\frac{|J|+1}{2}$ with probability less than $\frac{1-\sigma}{N}$. Hence, the probability of n being congruent to any element of $X - \{n\}$ modulo half of J's primes (with multiplicity) is less than $\frac{1-\sigma}{N}$. The union bound can now be employed to show that the majority of J's primes will separate every element of $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ from the (other) elements of X with probability at least σ. □

Lemma 2 creates a (k, σ)-majority selective collection of subsets, $\mathcal{S}$, which is a *multiset*: some $\mathcal{S}_j$-prime(s) may be selected multiple times. If this occurs, we treat the discrete Fourier transform related to each multiply-selected prime as multiple transforms for counting purposes only. That is, we only calculate one DFT per multiply-selected prime, perform one frequency identification step in Algorithm 2 of Section 5 per multiply-selected prime, etc.. However, we count the *results* of these calculations multiple times for the purposes of coefficient estimation (e.g., for lines 23 through 30 of Algorithm 2). Finally, note that in Lemma 2 we may set $K \geq C \cdot k\lfloor \log_{p_q} N \rfloor$ for any integer $C \in \{3, 4, 5, \dots\}$ and still obtain similar results. There is no particular reason for choosing $C = 7$ in Lemma 2 above.

We conclude this section by bounding the number of subsets contained in our k-majority and (k, σ)-majority selective collections. These subset bounds will ultimately provide us with sampling and runtime bounds for our Fourier algorithms. The following lemma is easily proved using results from [32].

Lemma 3. *Choose q so that p_q is the smallest prime $\geq k$. If $\mathcal{S}$ is a k-majority selective collection of subsets created as per Lemma 1, then $|\mathcal{S}|$ is $\Theta\left(k^2 \cdot \log_k^2 N \cdot \log(k \log N)\right)$. If $\mathcal{S}$ is a $\left(k, 1 - \frac{1}{N^{O(1)}}\right)$-majority selective collection of subsets created as per Lemma 2, then $|\mathcal{S}|$ is $O\left(k \cdot \log_k N \cdot \log(k \log N) \cdot \log N\right)$.*

Proof:

Suppose $\mathcal{S}$ is a k-majority selective collection of subsets created as per Lemma 1. In this case

$$|\mathcal{S}| = \sum_{j=0}^{K} p_{q+j}.$$

It follows from results in [32] that

$$\sum_{j=0}^{K} p_{q+j} = \frac{p_{q+K+1}^2}{2 \ln p_{q+K+1}} \cdot \left(1 + O\left(\frac{1}{\ln p_{q+K+1}}\right)\right) - \frac{p_q^2}{2 \ln p_q} \cdot \left(1 + O\left(\frac{1}{\ln p_q}\right)\right). \tag{15}$$

Furthermore, since p_q is the smallest prime $\geq k$, the Prime Number Theorem (see [47]) tells us that

$$q = \frac{k}{\ln k}\left(1 + O\left(\frac{1}{\ln k}\right)\right)$$

and

$$p_q = k\left(1 + O\left(\frac{\ln \ln k}{\ln k}\right)\right).$$

Thus, if we use the smallest possible value for K (i.e., $K = 2k\lfloor \log_{p_q} N \rfloor$), we can see that $q + K + 1$ is $\Theta\left(k \cdot \log_k N\right)$. Applying the Prime Number Theorem once more we have that

$$p_{q+K+1} = \Theta\left(k \cdot \log_k N \cdot \log\left(k \cdot \log N\right)\right). \tag{16}$$

Utilizing Equation 15 now reveals that

$$|\mathcal{S}| = \sum_{j=0}^{K} p_{q+j} = \Theta\left(k^2 \cdot \log_k^2 N \cdot \log(k \cdot \log N)\right).$$

If $\mathcal{S}$ is $\left(k, 1 - \frac{1}{N^{O(1)}}\right)$-majority selective, the stated $|\mathcal{S}|$ bound follows from Lemma 2 combined with Equation 16. In this case $\mathcal{S}$ consists of $O(\log N)$ randomly selected $\mathcal{S}_j$-sets, each with cardinality less than p_{q+K+1}. After noting that Equation 16 still holds if $K = 7k\lfloor \log_{p_q} N \rfloor$, we can see that

$$|\mathcal{S}| = O\left(p_{q+K+1} \cdot \log N\right) = O\left(k \cdot \log_k N \cdot \log\left(k \cdot \log N\right) \cdot \log N\right).$$

The theorem follows. □

Let $\alpha \in (0, 1)$ be a constant, and suppose that $k = \Theta(N^\alpha)$ so that $\log_k N$ is $O(1)$. In this case, we have a construction for k-majority selective collections, $\mathcal{S}$, with $|\mathcal{S}| = \Theta\left(k^2 \cdot \log N\right)$. Furthermore, we have a construction for $\left(k, 1 - \frac{1}{N^{O(1)}}\right)$-majority selective collections, $\mathcal{S}$, with $|\mathcal{S}| = O\left(k \cdot \log^2 N\right)$.

Algorithm 1 SUPERLINEAR APPROXIMATE

```
1:  Input: Signal pointer $f$, integers $k \leq B \leq N$
2:  Output: $\hat{\mathbf{R}}^{s}$, a sparse representation for $\hat{f}$
3:  Initialize $\hat{\mathbf{R}}^{s} \leftarrow \emptyset$
4:  Set $K = 2B \left\lfloor \log_{p_q} N \right\rfloor$, $q$ so that $p_{q-1} < B \leq p_q$
5:  for $j$ from 0 to $K$ do
6:      $\mathbf{A}_{p_{q+j}} \leftarrow f(0), f\left(\frac{2\pi}{p_{q+j}}\right), \ldots, f\left(\frac{2\pi(p_{q+j}-1)}{p_{q+j}}\right)$
7:      $\widehat{\mathbf{A}_{p_{q+j}}} \leftarrow \mathbf{DFT}[\mathbf{A}_{p_{q+j}}]$
8:  end for
9:  for $\omega$ from $1 - \left\lceil \frac{N}{2} \right\rceil$ to $\left\lfloor \frac{N}{2} \right\rfloor$ do
10:     $\mathbb{R}\mathrm{e}\{C_\omega\} \leftarrow$ median of multiset $\left\{\mathbb{R}\mathrm{e}\left\{\widehat{\mathbf{A}_{p_{q+j}}}(\omega \bmod p_{q+j})\right\} \,\middle|\, 0 \leq j \leq K\right\}$
11:     $\mathbb{I}\mathrm{m}\{C_\omega\} \leftarrow$ median of multiset $\left\{\mathbb{I}\mathrm{m}\left\{\widehat{\mathbf{A}_{p_{q+j}}}(\omega \bmod p_{q+j})\right\} \,\middle|\, 0 \leq j \leq K\right\}$
12: end for
13: $\hat{\mathbf{R}}^{s} \leftarrow (\omega, C_\omega)$ entries for $k$ largest magnitude $C_\omega$'s
```

4 Superlinear-Time Fourier Algorithms

For the remainder of the paper we will assume that $f : [0, 2\pi] \mapsto \mathbb{C}$ has the property that $\hat{f} \in l^1$. Our goal is to identify k of the most energetic frequencies in $\hat{f}$ (i.e., the first k entries in a valid ordering of $\hat{f}$ as in Equation 5) and then estimate their Fourier coefficients. Intuitively, we want f to be a continuous function which is dominated by a small number of energetic frequencies spread out over a large bandwidth. In this scenario our algorithms will allow us to ignore f's bandwidth and instead sample at a rate primarily dependent on the number of energetic frequencies present in f's Fourier spectrum.

Let $C \geq 1$ be a constant (to be specified later) and set

$$\epsilon = \frac{|\hat{f}(\omega_k)|}{C}. \tag{17}$$

Now, let B be the smallest integer such that

$$\sum_{b=B+1}^{\infty} |\hat{f}(\omega_b)| \leq \frac{\epsilon}{2}. \tag{18}$$

Note that ω_B is defined to be the last possible significant frequency $\left(\text{i.e., with energy} > \text{a fraction of } |\hat{f}(\omega_k)|\right)$. We will assume below that N is chosen large enough so that

$$\Omega = \{\omega_1, \ldots, \omega_B\} \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]. \tag{19}$$

We expect to work with signals for which $k \leq B \ll N$. Later we will give specific values for C and B depending on k, the desired approximation error, and $\hat{f}$'s compressibility characteristics. For now we show that we can identify/approximate k of $\hat{f}$'s largest magnitude entries each to within ϵ-precision via Algorithm 1.

Algorithm 1 works by using the k-majority separating structure created by the aliased DFTs in line 7 to isolate $\hat{f}$'s significantly energetic frequencies. Every DFT which successfully separates a frequency ω_j from all the (other) members of Ω will provide a good $\left(\text{i.e., within } \frac{\epsilon}{2} \leq \frac{|\hat{\mathbf{A}}(\omega_k)|}{2}\right)$ coefficient estimate for ω_j. Frequency separation occurs because more than $\frac{1}{2}$ of our aliased DFT's won't collide any $n \in \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$ with any (other) member of Ω (see Lemma 1). At most $B \log_{p_q} N$ of the DFT calculations for any particular frequency can be significantly contaminated via collisions with Ω members. Therefore, we can take medians of the real/imaginary parts of the $2B \log_{p_q} N + 1$ DFT residues associated with each frequency coefficient as a good estimate of that frequency coefficient's real/imaginary parts. Since more than half of these measurements must be accurate, the medians will be accurate. In order to formalize this argument we need the following lemma.

Lemma 4. *Every C_ω calculated in lines 10 and 11 is such that $|\hat{f}(\omega) - C_\omega| \leq \epsilon$.*

Proof:

Suppose that C_ω is calculated by lines 10 and 11. Then, its real/imaginary part is given by the median of $K + 1$ estimates of $\hat{f}(\omega)$'s real/imaginary parts. Each of these estimates is calculated by

$$\widehat{\mathbf{A}_{p_{q+j}}}(h) = \frac{2\pi}{p_{q+j}} \sum_{m=0}^{p_{q+j}-1} f\left(\frac{2\pi m}{p_{q+j}}\right) e^{\frac{-2\pi \mathbb{i} hm}{p_{q+j}}} \tag{20}$$

for some $0 \leq j \leq K$, $0 \leq h < p_{q+j}$. Via aliasing each estimate reduces to

$$\widehat{\mathbf{A}_{p_{q+j}}}(h) = \frac{2\pi}{p_{q+j}} \sum_{m=0}^{p_{q+j}-1} f\left(\frac{2\pi m}{p_{q+j}}\right) e^{\frac{-2\pi \mathbb{i} hm}{p_{q+j}}} = \frac{2\pi}{p_{q+j}} \sum_{m=0}^{p_{q+j}-1} \left(\frac{1}{2\pi} \sum_{\rho=-\infty}^{\infty} \hat{f}(\rho) e^{\frac{2\pi \mathbb{i} \rho m}{p_{q+j}}}\right) e^{\frac{-2\pi \mathbb{i} hm}{p_{q+j}}} \tag{21}$$

$$= \sum_{\rho=-\infty}^{\infty} \hat{f}(\rho) \left(\frac{1}{p_{q+j}} \sum_{m=0}^{p_{q+j}-1} e^{\frac{2\pi \mathbb{i}(\rho-h)m}{p_{q+j}}}\right) = \sum_{\rho \equiv h \bmod p_{q+j}} \hat{f}(\rho) \tag{22}$$

$$= \left\langle \chi_{S_{j,h}}, \hat{f} \cdot \chi_{\left(-\lceil \frac{N}{2} \rceil, \lfloor \frac{N}{2} \rfloor\right]} \right\rangle + \sum_{\rho \equiv h \bmod p_{q+j}, \rho \notin \left(-\lceil \frac{N}{2} \rceil, \lfloor \frac{N}{2} \rfloor\right]} \hat{f}(\rho). \tag{23}$$

Thus, by Lemma 1 and Equations 18 and 19, more than half of our $\hat{f}(\omega)$ estimates will have

$$\left|\hat{f}(\omega) - \widehat{\mathbf{A}_{p_{q+j}}}(\omega \bmod p_{q+j})\right| \leq \sum_{\rho \notin \Omega} \left|\hat{f}(\rho)\right| \leq \frac{\epsilon}{2}.$$

It follows that taking medians as per lines 10 and 11 will result in the desired ϵ-accurate estimate for $\hat{f}(\omega)$. □

It is natural to wonder whether lines 10 and 11 in Algorithm 1 can be replaced by a single median of the absolute values of the proper $\widehat{\mathbf{A}_{p_{q+j}}}$ entries. In fact, an argument similar to one establishing Lemma 4 above can be used to show that such a single median will indeed produce an accurate estimate of each frequency coefficient's magnitude. However, it is theoretically possible that the selected median-magnitude $\widehat{\mathbf{A}_{p_{q+j}}}$ entry would have a highly inaccurate phase. For example, the selected median-magnitude coefficient value for a frequency ω could be an entry from a DFT that collided ω with a member of $\Omega - \{\omega\}$ having a coefficient equal to $-2 \cdot \hat{f}(\omega)$. In this case the selected median-magnitude value would have the correct magnitude, but the phase would be off by π radians (i.e., we would recover the negative of the correct value). Thus, if we want to estimate each frequency coefficient (and not just the coefficient magnitudes) it appears to be necessary to separately consider both the real and imaginary parts of each frequency coefficient.

Finally, we note that the condition on q, that $p_{q-1} < B \leq p_q$, in Algorithm 1 is not strictly necessary. We assume it here so that the runtime and sampling complexities for Algorithm 1 can be derived in terms of B and N using Lemma 3 and analytic number theoretic results from [32]. See Section 7 for more on implementation details, relaxing conditions on q, etc. The following Theorem presents itself.

Theorem 2. *Let $\hat{\boldsymbol{R}}_{\text{opt}}$ be a k-optimal Fourier representation for the Fourier transform of our input function f. Then, the k-term representation $\hat{\boldsymbol{R}}^{\text{s}}$ returned from Algorithm 1 is such that $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \frac{9k \cdot |\hat{f}(\omega_k)|^2}{C}$. Furthermore, Algorithm 1's runtime is $O\left(N \cdot B \cdot \frac{\log^2 N \cdot \log^2(B \log N)}{\log^2 B}\right)$. The number of f samples used is $\Theta\left(B^2 \cdot \log_B^2 N \cdot \log(B \log N)\right)$.*

Proof:

Choose any $b \in (0, k]$. Using Lemma 4 we can see that the only way some $\omega_b \notin \hat{\mathbf{R}}^s_B$ is if there exists some associated $b' \in (k, N]$ so that $\omega_{b'} \in \hat{\mathbf{R}}^s$ and

$$|\hat{f}(\omega_k)| + \epsilon \geq |\hat{f}(\omega_{b'})| + \epsilon \geq |C_{\omega_{b'}}| \geq |C_{\omega_b}| \geq |\hat{f}(\omega_b)| - \epsilon \geq |\hat{f}(\omega_k)| - \epsilon.$$

In this case we will have $2\epsilon > |\hat{f}(\omega_b)| - |\hat{f}(\omega_{b'})| \geq 0$ so that

$$|\hat{f}(\omega_{b'})|^2 + 4\epsilon\left(\epsilon + |\hat{f}(\omega_k)|\right) \geq |\hat{f}(\omega_{b'})|^2 + 4\epsilon\left(\epsilon + |\hat{f}(\omega_{b'})|\right) \geq |\hat{f}(\omega_b)|^2. \tag{24}$$

Now using Lemma 4 we can see that

$$\|\hat{f} - \hat{\mathbf{R}}\|^2 = \sum_{(\omega,\cdot)\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega)|^2 + \sum_{(\omega,C_\omega)\in\hat{\mathbf{R}}^s} |\hat{f}(\omega) - C_\omega|^2 \leq \sum_{(\omega,\cdot)\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega)|^2 + k \cdot \epsilon^2.$$

Furthermore, we have

$$k \cdot \epsilon^2 + \sum_{(\omega,\cdot)\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega)|^2 = k \cdot \epsilon^2 + \sum_{b\in(0,k],\ \omega_b\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega_b)|^2 + \sum_{b'\in(k,N],\ \omega_{b'}\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega_{b'})|^2.$$

Using observation (24) above we can see that this last expression is bounded above by

$$k \cdot (5\epsilon^2 + 4\epsilon|\hat{f}(\omega_k)|) + \sum_{b'\in(k,N],\ \omega_{b'}\in\hat{\mathbf{R}}^s} |\hat{f}(\omega_{b'})|^2 + \sum_{b'\in(k,N],\ \omega_{b'}\notin\hat{\mathbf{R}}^s} |\hat{f}(\omega_{b'})|^2 \leq \|\hat{f} - \hat{\mathbf{R}}_{\text{opt}}\|_2^2 + k \cdot (5\epsilon^2 + 4\epsilon|\hat{f}(\omega_k)|).$$

Substituting for ϵ (see Equation 17) gives us our result. Namely,

$$k \cdot (5\epsilon^2 + 4\epsilon|\hat{f}(\omega_k)|) = \frac{k|\hat{f}(\omega_k)|^2}{C}\left(\frac{5}{C} + 4\right) \leq \frac{9k|\hat{f}(\omega_B)|^2}{C}.$$

We conclude by providing sampling/runtime bounds for Algorithm 1. Lines 5 through 8 take $O\left(\sum_{j=0}^{K} p_{q+j} \log p_{q+j}\right)$ time if the $K + 1$ DFTs are each computed using the Chirp z-Transform [3, 48]. Furthermore, Lemma 5 and Equation 8 from [32] reveal that

$$\sum_{j=0}^{K} p_{q+j} \log p_{q+j} = \Theta\left(B^2 \cdot \frac{\log^2 N \cdot \log^2(B \log N)}{\log^2 B}\right).$$

Continuing, each line 10 and 11 median calculation can be performed using a $O(K \cdot \log K)$-time sorting algorithm (see [10]). Thus, lines 9 through 13 require $O\left(N \cdot B \log_B N \cdot \log(B \log N)\right)$ time in total. Combining the runtime bounds for lines 5 through 8 and lines 9 through 13 gives us the stated runtime result. The sampling complexity follows directly from Lemma 3. □

It is not difficult to see that the proofs of Lemma 4 and Theorem 2 still hold using the (k, σ)-majority selective properties of randomly chosen primes. In particular, if we run Algorithm 1 using randomly chosen primes along the lines of Lemma 2 then Theorem 2 will still hold whenever the primes behave in a majority selective fashion. The only change required to Algorithm 1 is that we compute only a random subset of the DFTs in lines 5 through 8. We have the following corollary.

Corollary 1. *Let $\hat{\boldsymbol{R}}_{\text{opt}}$ be a k-optimal Fourier representation for the Fourier transform of our input function f. If we run Algorithm 1 using $O\left(\log\left(\frac{N}{1-\sigma}\right)\right)$ randomly selected primes along the lines of Lemma 2, then with probability at least σ we will obtain a k-term representation $\hat{\boldsymbol{R}}^s$ having $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \frac{9k\cdot|\hat{f}(\omega_k)|^2}{C}$. The runtime will be $O\left(N \cdot \log_B N \cdot \log\left(\frac{N}{1-\sigma}\right) \cdot \log^2\left(B \log\left(\frac{N}{1-\sigma}\right)\right)\right)$. The number of f samples will be $O\left(B \cdot \log_B N \cdot \log(B \log N) \cdot \log\left(\frac{N}{1-\sigma}\right)\right)$.*

It has been popular in the compressed sensing literature to consider the recovery of k-frequency superpositions (see [35] and references therein). Suppose we have

$$f(x) = \sum_{b=1}^{k} C_b \cdot e^{\mathbb{i}\omega_b x} \text{ for all } x \in [0, 2\pi],\ \Omega = \{\omega_1, \ldots, \omega_k\} \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]. \tag{25}$$

Setting $B = k$ and $C = 1$ is then sufficient to guarantee that $\sum_{b=B+1}^{\infty} |\hat{f}(\omega_b)| = 0$. Theorem 2 now tells us that Algorithm 1 will perfectly reconstruct f. We quickly obtain the final result of this section.

Corollary 2. *Suppose f is a k-frequency superposition. Then, Algorithm 1 can exactly recover f in $O\left(N \cdot k \cdot \frac{\log^2 N \cdot \log^2 (k \log N)}{\log^2 k}\right)$ time. The number of f samples used is $\Theta\left(k^2 \cdot \log_k^2 N \cdot \log(k \log N)\right)$. If we run Algorithm 1 using $O\left(\log\left(\frac{N}{1-\sigma}\right)\right)$ randomly selected primes along the lines of Lemma 2, then we will exactly recover f with probability at least σ. In this case the runtime will be $O\left(N \cdot \log_k N \cdot \log\left(\frac{N}{1-\sigma}\right) \cdot \log^2\left(k \log\left(\frac{N}{1-\sigma}\right)\right)\right)$. The number of f samples will be $O\left(k \cdot \log_k N \cdot \log(k \log N) \cdot \log\left(\frac{N}{1-\sigma}\right)\right)$.*

As before, let $\alpha \in (0, 1)$ be a constant and suppose that $k = \Theta(N^\alpha)$. Furthermore, let $\sigma = 1 - \frac{1}{N^{O(1)}}$. Corollary 2 implies that our deterministic Algorithm 1 exactly recovers k-frequency superpositions using $O(k^2 \log N)$ samples. If randomly selected primes are used, then Algorithm 1 can exactly reconstruct k-frequency superpositions with probability $1 - \frac{1}{N^{O(1)}}$ using $O(k \log^2 N)$ samples. In this case the sampling complexity of the randomized variant of Algorithm 1 is within a logarithmic factor of the best known Fourier sampling bounds concerning high probability exact recovery of superpositions [5, 35]. This is encouraging given Algorithm 1's simplicity. Of greater interest for our purposes here, however, is that Algorithm 1 can be easily modified to run in sublinear time.

5 Sublinear-Time Fourier Algorithms

In order to reduce the runtime of Algorithm 1 we will replace lines 9 through 13 with a more general version of the example deterministic recovery approach discussed in Section 1.1.3. This new recovery approach will take advantage of the combinatorial properties of line 7's $K+1$ aliased DFTs (i.e., see Definition 1 and Lemma 1) as follows: Suppose we can use the CRT to identify all of the k most energetic frequencies that are isolated from all the other elements of Ω by any given line 7 DFT. Then, by the k-majority selective properties of our $K + 1$ DFTs, we will be guaranteed to identify all of the k most energetic frequencies more than $\frac{K+1}{2}$ times each. Therefore, collecting all frequencies recovered by more than half of line 7's DFTs will give us the k most energetic Ω frequencies (along with some possibly 'junk frequencies'). The 'junk' can be discarded, however, by limitedly applying our existing coefficient estimation method (i.e., lines 10 and 11) to the identified potentially-energetic frequencies. Only truly energetic frequencies will yield large magnitude coefficient estimates by Lemma 4.

Finally, note that only $O(K \log K)$ potentially energetic frequencies can be identified more than $\frac{K+1}{2}$ times via line 7's DFTs. Thus, our formally superlinear-time loop (lines 9 - 12) will be sublinearized. Instead of applying lines 10 and 11 to *every* frequency, we will only apply them to the at most $O(K \log K)$ potentially-energetic frequencies we identify. See Algorithm 2 for the sublinear-time algorithm obtained by modifying Algorithm 1 as outlined above.

Let m be the smallest integer such that

$$\prod_{l=0}^{m} p_l \geq \frac{N}{B}. \tag{26}$$

The following lemma establishes the correctness of Algorithm 2's energetic frequency identification procedure.

Lemma 5. *Lines 11 through 22 of Algorithm 2 are guaranteed to recover all valid $\omega_1, \ldots, \omega_k$ (i.e., all ω with $|\hat{\boldsymbol{A}}(\omega)|_2 \geq |\hat{\boldsymbol{A}}(\omega_k)|_2$ - there may be $> k$ such entries) more than $\frac{K}{2}$ times. Hence, despite line 25, an entry for all such $\omega_b, 1 \leq b \leq k$, will be added to $\hat{\boldsymbol{R}}^{\mathrm{s}}$ in line 31.*

Algorithm 2 SUBLINEAR APPROXIMATE

```
1:  Input: Signal pointer $f$, integers $m, k \leq B \leq N$
2:  Output: $\hat{\mathbf{R}}^{s}$, a sparse representation for $\hat{f}$
3:  Initialize $\hat{\mathbf{R}}^{s} \leftarrow \emptyset$
4:  Set $K = 2B \left\lfloor \log_{p_q} N \right\rfloor$, $q$ so that $p_{q-1} \leq \max(B, p_m) < p_q$
5:  for $j$ from 0 to $K$ do
6:      for $l$ from 0 to $m$ do
7:          $\mathbf{A}_{p_l \cdot p_{q+j}} \leftarrow f(0), f\left(\frac{2\pi}{p_l \cdot p_{q+j}}\right), \ldots, f\left(\frac{2\pi(p_l \cdot p_{q+j}-1)}{p_l \cdot p_{q+j}}\right)$
8:          $\widehat{\mathbf{A}_{p_l \cdot p_{q+j}}} \leftarrow \mathbf{DFT}[\mathbf{A}_{p_l \cdot p_{q+j}}]$
9:      end for
10: end for
                    ENERGETIC FREQUENCY IDENTIFICATION
11: for $j$ from 0 to $K$ do
12:     $\hat{A}_{\text{sort}} \leftarrow$ Sort $\widehat{\mathbf{A}_{p_0 \cdot p_{q+j}}}$ by magnitude (i.e., $b^{\text{th}}$ largest magnitude entry in $\hat{A}_{\text{sort}}(b)$)
13:     for $b$ from 1 to $B$ do
14:         $r_{0,b} \leftarrow$ index of $\widehat{\mathbf{A}_{p_0 \cdot p_{q+j}}}$'s $b^{\text{th}}$ largest magnitude entry $\left(\text{i.e., } \hat{A}_{\text{sort}}(b)\text{'s associated residue mod } p_{q+j}\right)$
15:         for $l$ from 1 to $m$ do
16:             $t_{\min} \leftarrow \min_{t \in [0, p_l)} \left| \hat{A}_{\text{sort}}(b) - \widehat{\mathbf{A}_{p_l \cdot p_{q+j}}}(t \cdot p_{q+j} + r_{0,b}) \right|$
17:             $r_{l,b} \leftarrow \left(r_{0,b} + t_{\min} \cdot p_{q+j}\right) \bmod p_l$
18:         end for
19:         Construct $\omega_{j,b}$ from $r_{0,b}, \ldots, r_{m,b}$ via modular arithmetic
20:     end for
21: end for
22: Sort $\omega_{j,b}$'s maintaining duplicates and set $C(\omega_{j,b})$ = the number of times $\omega_{j,b}$ was constructed via line 19
                    COEFFICIENT ESTIMATION
23: for $j$ from 1 to $K$ do
24:     for $b$ from 1 to $B$ do
25:         if $C(\omega_{j,b}) > \frac{K}{2}$ then
26:             $\mathbb{Re}\left\{C_{\omega_{j,b}}\right\} \leftarrow$ median of multiset $\left\{\mathbb{Re}\left\{\widehat{\mathbf{A}_{p_m \cdot p_{q+h}}}(\omega_{j,b} \bmod p_m \cdot p_{q+h})\right\} \,\middle|\, 0 \leq h \leq K\right\}$
27:             $\mathbb{Im}\left\{C_{\omega_{j,b}}\right\} \leftarrow$ median of multiset $\left\{\mathbb{Im}\left\{\widehat{\mathbf{A}_{p_m \cdot p_{q+h}}}(\omega_{j,b} \bmod p_m \cdot p_{q+h})\right\} \,\middle|\, 0 \leq h \leq K\right\}$
28:         end if
29:     end for
30: end for
31: $\hat{\mathbf{R}}^{s} \leftarrow (\omega_{j,b}, C_{\omega_{j,b}})$ entries for $k$ largest magnitude $C_{\omega_{j,b}}$'s
```

Proof:

Fix $\tilde{b} \in [1, k]$. By Lemma 1 we know that there exist more than $\frac{K}{2}$ p_{q+j}-primes that isolate $\omega_{\tilde{b}}$ from all of $\Omega - \{\omega_{\tilde{b}}\}$. Denote these primes by

$$p_{j_1}, p_{j_2}, \ldots, p_{j_{K'}}, \quad \frac{K}{2} < K' \leq K.$$

We next show, for each $k' \in [1, K']$, that we get $\widehat{\mathbf{A}_{p_0 \cdot p_{j_{k'}}}}(\omega_{\tilde{b}} \bmod p_{j_{k'}})$ as one of the B largest magnitude entries found in line 12. Choose any $k' \in [1, K']$. Using Equations 17 and 18 we can see that

$$\frac{\epsilon}{2} \leq |\hat{f}(\omega_k)| - \sum_{b'=B+1}^{\infty} |\hat{f}(\omega_{b'})| \leq |\hat{f}(\omega_{\tilde{b}})| - \left| \sum_{b' \notin \Omega,\ \omega_{b'} \equiv \omega_{\tilde{b}}} \hat{f}(\omega_{b'}) \right| \leq \left| \widehat{\mathbf{A}_{p_0 \cdot p_{j_{k'}}}}(\omega_{\tilde{b}} \bmod p_{j_{k'}}) \right|.$$

We also know that the $(B+1)^{\text{st}}$ largest magnitude entry of $\widehat{\mathbf{A}_{p_0 \cdot p_{j_{k'}}}}$ must be $\leq \frac{\epsilon}{2}$. Hence, for every $k' \in [1, K']$ we are guaranteed to execute lines 13-20 with an $r_{0,b} = \omega_{\tilde{b}} \bmod p_{j_{k'}}$.

Next, we will show that all the residues required in order to reconstruct $\omega_{\tilde{b}}$ by the CRT in line 19 will be found. Choose any $l \in [1, m]$ and set

$$\bar{\Omega}' = \left\{ \omega_{b'} \,\middle|\, \omega_{b'} \notin \Omega,\ \omega_{b'} \equiv \omega_{\tilde{b}} \bmod p_{j_{k'}},\ \omega_{b'} \not\equiv \omega_{\tilde{b}} \bmod p_l p_{j_{k'}} \right\}.$$

Line 16 inspects all possible residues of $\omega_{\tilde{b}} \bmod p_l p_{j_{k'}}$ since

$$\omega_{\tilde{b}} \equiv h \bmod p_{j_{k'}} \longrightarrow \omega_{\tilde{b}} \equiv h + t \cdot p_{j_{k'}} \bmod p_l p_{j_{k'}}$$

for some $t \in [0, p_l)$. To see that t_{min} will be chosen correctly, we note first that

$$\left| \widehat{\mathbf{A}_{p_0 \cdot p_{j_{k'}}}} (\omega_{\tilde{b}} \bmod p_{j_{k'}}) - \widehat{\mathbf{A}_{p_l \cdot p_{j_{k'}}}} (\omega_{\tilde{b}} \bmod p_l p_{j_{k'}}) \right| \leq \sum_{\omega_{b'} \in \bar{\Omega}'} |\hat{f}(\omega_{b'})| \leq \frac{\epsilon}{2} \leq |\hat{f}(\omega_k)| - \sum_{b'=B+1}^{\infty} |\hat{f}(\omega_{b'})|.$$

Furthermore, setting $r_{0,b} = \omega_{\tilde{b}} \bmod p_{j_{k'}}$ and

$$\tilde{\Omega}' = \left\{ \omega_{b'} \,\middle|\, \omega_{b'} \notin \Omega,\ \omega_{b'} \equiv \omega_{\tilde{b}} \bmod p_{j_{k'}},\ \omega_{b'} \not\equiv (r_{0,b} + tp_{j_{k'}}) \bmod p_{j_{k'}} p_l \text{ for some } t \text{ with } (r_{0,b} + tp_{j_{k'}}) \not\equiv \omega_{\tilde{b}} \bmod p_l p_{j_{k'}} \right\},$$

we have

$$|\hat{f}(\omega_k)| - \sum_{b'=B+1}^{\infty} |\hat{f}(\omega_{b'})| \leq |\hat{f}(\omega_{\tilde{b}})| - \left| \sum_{\omega_{b'} \in \tilde{\Omega}'} \hat{f}(\omega_{b'}) \right| \leq \left| \widehat{\mathbf{A}_{p_0 \cdot p_{j_{k'}}}} (\omega_{\tilde{b}} \bmod p_{j_{k'}}) - \widehat{\mathbf{A}_{p_l \cdot p_{j_{k'}}}} \left((r_{0,b} + tp_{j_{k'}}) \not\equiv \omega_{\tilde{b}} \bmod p_l p_{j_{k'}} \right) \right|.$$

Hence, lines 16 and 17 will indeed select the correct residue for ω_b modulo p_l. And, line 19 will correctly reconstruct ω_b at least $K' > \frac{K}{2}$ times. □

Using Lemma 5 along with Lemma 4 and Theorem 2 we obtain the following Theorem concerning Algorithm 2. The sampling and runtime bounds are computed in [32].

Theorem 3. *Let $\hat{\boldsymbol{R}}_{\text{opt}}$ be a k-optimal Fourier representation for the Fourier transform of our input function f. Then, the k-term representation $\hat{\boldsymbol{R}}^{\text{s}}$ returned from Algorithm 2 is such that $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \frac{9k \cdot |\hat{f}(\omega_k)|^2}{C}$. Furthermore, Algorithm 2's runtime is $O\left(B^2 \cdot \frac{\log^2 N \cdot \log^2(B \log N) \cdot \log^2 \frac{N}{B}}{\log^2 B \cdot \log\log \frac{N}{B}}\right)$. The number of f samples used is $O\left(B^2 \cdot \frac{\log^2 N \cdot \log(B \log N) \cdot \log^2 \frac{N}{B}}{\log^2 B \cdot \log\log \frac{N}{B}}\right)$.*

Also, as above, if we run Algorithm 2 using randomly chosen p_{q+j}-primes along the lines of Lemma 2 then Theorem 3 will still hold whenever the p_{q+j}-primes behave in a majority selective fashion. We have the following corollary.

Corollary 3. *Let $\hat{\boldsymbol{R}}_{\text{opt}}$ be a k-optimal Fourier representation for the Fourier transform of our input function f. If we run Algorithm 2 using $O\left(\log\left(\frac{N}{1-\sigma}\right)\right)$ randomly selected p_{q+j}-primes along the lines of Lemma 2, then with probability at least σ we will obtain a k-term representation $\hat{\boldsymbol{R}}^{\text{s}}$ having $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \frac{9k \cdot |\hat{f}(\omega_k)|^2}{C}$. The runtime will be $O\left(B \cdot \frac{\log N \cdot \log\left(\frac{N}{1-\sigma}\right) \cdot \log^2\left(B \log\left(\frac{N}{1-\sigma}\right)\right) \cdot \log^2 \frac{N}{B}}{\log B \cdot \log\log \frac{N}{B}}\right)$. The number of f samples will be $O\left(B \cdot \frac{\log^2\left(\frac{N}{1-\sigma}\right) \cdot \log(B \log N) \cdot \log^2 \frac{N}{B}}{\log B \cdot \log\log \frac{N}{B}}\right)$.*

Let $\alpha \in (0,1)$ be a constant and suppose that $k = \Theta(N^\alpha)$. Furthermore, suppose that $\sigma = 1 - \frac{1}{N^{O(1)}}$. Theorem 3 tells us that our sublinear-time deterministic Algorithm 2 exactly recovers k-frequency superpositions in $O\left(k^2 \cdot \frac{\log^4 N}{\log\log N}\right)$ time using $O\left(k^2 \cdot \frac{\log^3 N}{\log\log N}\right)$ samples. If randomly selected p_{q+j}-primes are used then Algorithm 2 can exactly reconstruct k-frequency superpositions with probability $1 - \frac{1}{N^{O(1)}}$ in $O\left(k \cdot \frac{\log^5 N}{\log\log N}\right)$ time using $O\left(k \cdot \frac{\log^4 N}{\log\log N}\right)$ samples. It is worth noting here that previous randomized sublinear-time Fourier results [24, 25] do not yield exact reconstructions of sparse superpositions in this manner. They iteratively produce approximate solutions which converge to the true superposition in the limit.

We are now ready to give sublinear-time results concerning functions with compressible Fourier coefficients. For the remainder of this paper we will assume that our input function $f : [0, 2\pi] \mapsto \mathbb{C}$ has both (i) an integrable p^{th} derivative, and (ii) $f(0) = f(2\pi), f'(0) = f'(2\pi), \ldots, f^{(p-2)}(0) = f^{(p-2)}(2\pi)$ for some $p > 1$. Standard Fourier coefficient bounds then imply that $\hat{f}$ is a (c, p)-compressible ∞-length signal for some $c \in \mathbb{R}^+$ [22, 4]. Before applying Theorem 3 we will determine Algorithm 2's B and Equation 17's C variables based on the desired Fourier representation's size and accuracy. Moving toward that goal, we note that since $\hat{f}$ is algebraically compressible, we have

$$\frac{9k \cdot |\hat{f}(\omega_k)|^2}{C} \leq \frac{9c^2 \cdot k^{1-2p}}{C} = \Theta\left(\frac{2p-1}{C}\right) \|C_k^{\text{opt}}\|_2^2. \tag{27}$$

Thus, in order to achieve multiplicative accuracy δ (see Equation 11), we should set $C = \Theta\left(\frac{2p-1}{\delta}\right)$ and choose B so that

$$\sum_{b=B+1}^{\infty} |\hat{f}(\omega_b)| < \frac{c \cdot B^{1-p}}{p-1} \leq \frac{|\hat{f}(\omega_k)|}{2C} = O\left(\frac{c\delta}{2p-1} \cdot k^{-p}\right). \tag{28}$$

Solving, we get that

$$B = \Omega\left(\left(\frac{p-1}{2p-1}\right)^{\frac{1}{1-p}} \delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}}\right).$$

Applying Theorem 3 gives us the runtime and number of required measurements for Algorithm 2. We obtain the following Corollary.

Corollary 4. *Suppose $f : [0, 2\pi] \mapsto \mathbb{C}$ has both (i) an integrable p^{th} derivative, and (ii) $f(0) = f(2\pi), \ldots, f^{(p-2)}(0) = f^{(p-2)}(2\pi)$ for some fixed $p > 1$. Furthermore, assume that $\hat{f}$'s $B = O\left(\delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}}\right)$ largest magnitude frequencies all belong to $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$. Fix accuracy parameter $\delta \in \mathbb{R}^+$. Then, we may use Algorithm 2 to return a k-term sparse Fourier representation, $\hat{\boldsymbol{R}}^{\text{s}}$, for $\hat{f}$ with $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \delta\|C_k^{\text{opt}}\|_2^2$ in $O\left(\delta^{\frac{2}{1-p}} k^{\frac{2p}{p-1}} \cdot \frac{\log^6 N}{\log^2 \frac{k^p}{\delta}}\right)$ time. The number of f samples used is $O\left(\delta^{\frac{2}{1-p}} k^{\frac{2p}{p-1}} \cdot \frac{\log^5 N}{\log^2 \frac{k^p}{\delta}}\right)$. If we run Algorithm 2 using $O\left(\log\left(\frac{N}{1-\sigma}\right)\right)$ randomly selected p_{q+j}-primes along the lines of Lemma 2, then with probability at least σ we will obtain a k-term representation $\hat{\boldsymbol{R}}^{\text{s}}$ having $\|\hat{f} - \hat{\boldsymbol{R}}\|_2^2 \leq \|\hat{f} - \hat{\boldsymbol{R}}_{\text{opt}}\|_2^2 + \delta\|C_k^{\text{opt}}\|_2^2$ in $O\left(\delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}} \cdot \frac{\log^6 N}{\log \frac{k^p}{\delta}}\right)$ time. The number of f samples used is $O\left(\delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}} \cdot \frac{\log^5 N}{\log \frac{k^p}{\delta}}\right)$.*

If $f : [0, 2\pi] \to \mathbb{C}$ is smooth (i.e., has infinitely many continuous derivatives on the unit circle where 0 is identified with 2π) it follows from Corollary 4 that Algorithm 2 can be used to find an δ-accurate, with $\delta = O\left(\frac{1}{N}\right)$, sparse k-term Fourier representation for $\hat{f}$ in $O(k^2 \log^6 N)$ time using $O(k^2 \log^5 N)$ measurements. If randomly selected p_{q+j}-primes are utilized, then Algorithm 2 can obtain a $O\left(\frac{1}{N}\right)$-accurate k-term Fourier representation for $\hat{f}$ with high probability in $O(k \log^6 N)$ time using $O(k \log^5 N)$ measurements. Similarly, standard results concerning the exponential decay of Fourier coefficients for functions with analytic extensions can be used to generate exponentially compressible Fourier results.

6 Discrete Fourier Results

Suppose we are provided with an array $\mathbf{A}$ containing N equally spaced samples from an unknown smooth function $f : [0, 2\pi] \to \mathbb{C}$ (i.e., f is $\mathbf{A}$'s band-limited interpolent). Then, it is standard to assume that

$$f(x) = \frac{1}{2\pi} \sum_{\omega=1-\lceil \frac{N}{2} \rceil}^{\lfloor \frac{N}{2} \rfloor} \widehat{\mathbf{A}}(\omega) \cdot e^{\mathbb{i}\omega \cdot x}, \quad x \in [0, 2\pi].$$

with

$$\mathbf{A}(j) = f\left(\frac{2\pi j}{N}\right),\ j \in [0, N).$$

We would like to use Algorithm 2 to find a sparse Fourier representation for $\hat{\mathbf{A}}$. Not having access to f directly, and restricting ourselves to sublinear-time approaches only, we have little recourse but to locally interpolate f around Algorithm 2's required samples.

For each required Algorithm 2 f-sample at $t = \frac{2\pi h}{p_{q+j} p_l}, h \in [0, p_{q+j} p_l)$, we may approximate $f(t)$ to within $O\left(N^{-2\kappa}\right)$ error by constructing 2 standard local Lagrangian interpolents (one real, one imaginary) around t using $\mathbf{A}$'s nearest 2κ entries [33]. These errors in f-samples can lead to errors of size $O\left(N^{-2\kappa} \cdot p_m p_{q+K} \log p_{q+K}\right)$ in each of the DFT entries in line 8 of Algorithm 2. However, as long as these errors are small enough (i.e., of size $O\left(\frac{c\delta}{2p-1} \cdot k^{-p}\right)$ in the (c,p)-compressible case) Theorem 3 and all related Section 5 results will still hold. Hence, using $2\kappa = O\left(\log\left(\frac{2p-1}{c\delta} \cdot k^p\right)\right)$ interpolation points per f-sample should be sufficient. We have the following lemma.

Lemma 6. *Let $\mathbf{A}$ be an N-length complex valued array and suppose that $\hat{\mathbf{A}}$ is (c,p)-compressible. Fix $\tilde{c} \in \mathbb{R}^+$. Using*

$$2\kappa = 2 \cdot \log_8 \left(\frac{\sqrt{2} \cdot k^p}{\tilde{c} \cdot \delta} \cdot \frac{2p^2 - p}{p - 1} \right)$$

interpolation points from $\mathbf{A}$ per f-evaluation will guarantee that every line 8 DFT entry from Algorithm 2 is calculated to within $\frac{\tilde{c} \cdot c\delta}{2p-1} \cdot k^{-p}$ precision.

Proof:

Fix $x \in [0, 2\pi]$ and $\kappa \in \left(0, \frac{N}{2}\right) \cap \mathbb{N}$. Let $j' = \left\lfloor x \cdot \frac{N}{2\pi} \right\rfloor$. We will form two interpolating polynomials of degree at most $2\kappa - 1$. The first polynomial, $p_R^x : \mathbb{R} \to \mathbb{R}$, will be formed using the 2κ points

$$\left(\frac{2\pi m}{N},\ \mathbb{Re}\left\{\mathbf{A}\left(m \bmod N\right)\right\}\right),\quad m \in (j' - \kappa, j' + \kappa] \cap \mathbb{Z}.$$

The second polynomial, $p_I^x : \mathbb{R} \to \mathbb{R}$, will be formed using the 2κ points

$$\left(\frac{2\pi m}{N},\ \mathbb{Im}\left\{\mathbf{A}\left(m \bmod N\right)\right\}\right),\quad m \in (j' - \kappa, j' + \kappa] \cap \mathbb{Z}.$$

Using standard results concerning polynomial interpolation error (see [33]) we can see that

$$\left|\mathbb{Re}\left\{f(x)\right\} - p_R^x(x)\right| \ \leq\ \frac{\|f^{(2\kappa)}\|_\infty}{(2\kappa)!} \cdot \prod_{m=1}^{\kappa} \left(\frac{m}{N}\right)^2 \ \leq\ \frac{1}{(2\kappa)!} \cdot \frac{\|\,\hat{\mathbf{A}}\,\|_1}{2\pi} \left(\frac{N}{2}\right)^{2\kappa} \cdot \prod_{m=1}^{\kappa} \left(\frac{m}{N}\right)^2 \ \leq\ \frac{\|\,\hat{\mathbf{A}}\,\|_1}{2\pi \cdot 4^\kappa} \cdot \frac{\prod_{m=1}^{\kappa} m^2}{(2\kappa)!} \ \leq\ \frac{\|\,\hat{\mathbf{A}}\,\|_1}{2\pi \cdot 8^\kappa}.$$

An analogous bound holds for $\left|\mathbb{Im}\left\{f(x)\right\} - p_I^x(x)\right|$. Therefore, we can see that 2κ samples from $\mathbf{A}$ are sufficient to calculate $f(x)$ to within an error of $\frac{\|\hat{\mathbf{A}}\|_1}{\sqrt{2}\pi \cdot 8^\kappa}$.

To conclude, we can see that approximating the $p_l \cdot p_{q+j}$ equally spaced DFT of f using these interpolated values will lead to errors of size

$$\left| \frac{2\pi}{p_l \cdot p_{q+j}} \cdot \sum_{m=0}^{p_l \cdot p_{q+j} - 1} e^{\frac{-2\pi \mathbb{i} h m}{p_l \cdot p_{q+j}}} f\left(\frac{2\pi m}{p_l \cdot p_{q+j}}\right) - \frac{2\pi}{p_l \cdot p_{q+j}} \cdot \sum_{m=0}^{p_l \cdot p_{q+j} - 1} e^{\frac{-2\pi \mathbb{i} h m}{p_l \cdot p_{q+j}}} \left(p_R^{\frac{2\pi m}{p_l \cdot p_{q+j}}} \left(\frac{2\pi m}{p_l \cdot p_{q+j}}\right) + \mathbb{i} \cdot p_I^{\frac{2\pi m}{p_l \cdot p_{q+j}}} \left(\frac{2\pi m}{p_l \cdot p_{q+j}}\right) \right) \right|$$

for each $h \in \left(-\left\lceil \frac{p_l \cdot p_{q+j}}{2} \right\rceil, \left\lfloor \frac{p_l \cdot p_{q+j}}{2} \right\rfloor\right]$. Using the work above we can see that this error is bounded above by

$$\frac{2\pi}{p_l \cdot p_{q+j}} \cdot \left| \sum_{m=0}^{p_l \cdot p_{q+j} - 1} e^{\frac{-2\pi \mathbb{i} h m}{p_l \cdot p_{q+j}}} \frac{\|\,\hat{\mathbf{A}}\,\|_1}{\sqrt{2}\pi \cdot 8^\kappa} \right| \leq \frac{\sqrt{2}}{8^\kappa} \cdot \|\,\hat{\mathbf{A}}\,\|_1 \ \leq\ \frac{\sqrt{2}c}{8^\kappa} \cdot \left(1 + \int_1^\infty b^{-p}\, db\right) \ \leq\ \frac{\sqrt{2}c}{8^\kappa} \cdot \left(\frac{p}{p-1}\right).$$

The result follows. □

Combining Lemma 6 with Corollary 4 from Section 5 yields our final corollary to Theorem 3.

Corollary 5. *Let* $\mathbf{A}$ *be an* N*-length complex valued array and suppose that* $\hat{\mathbf{A}}$ *is* (c,p)*-compressible for fixed* $c, p \in \mathbb{R}^+$*. Then, we may use Algorithm 2 to return a* k*-term sparse Fourier representation,* $\hat{\boldsymbol{R}}^{\mathrm{s}}$*, for* $\hat{\mathbf{A}}$ *with* $\|\hat{\mathbf{A}} - \hat{\boldsymbol{R}}\|_2^2 \le \|\hat{\mathbf{A}} - \hat{\boldsymbol{R}}_{\mathrm{opt}}\|_2^2 + \delta\|C_k^{\mathrm{opt}}\|_2^2$ *in* $O\left(\delta^{\frac{2}{1-p}} k^{\frac{2p}{p-1}} \cdot \frac{\log^6 N}{\log \frac{k^p}{\delta}}\right)$ *time. The number of samples used is* $O\left(\delta^{\frac{2}{1-p}} k^{\frac{2p}{p-1}} \cdot \frac{\log^5 N}{\log \frac{k^p}{\delta}}\right)$*. If we run Algorithm 2 using* $O\left(\log\left(\frac{N}{1-\sigma}\right)\right)$ *randomly selected* p_{q+j}*-primes along the lines of Lemma 2, then with probability at least* σ *we will obtain a* k*-term representation* $\hat{\boldsymbol{R}}^{\mathrm{s}}$ *satisfying* $\|\hat{\mathbf{A}} - \hat{\boldsymbol{R}}\|_2^2 \le \|\hat{\mathbf{A}} - \hat{\boldsymbol{R}}_{\mathrm{opt}}\|_2^2 + \delta\|C_k^{\mathrm{opt}}\|_2^2$ *in* $O\left(\delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}} \cdot \log^6 N\right)$ *time. The number of* $\mathbf{A}$ *samples used is* $O\left(\delta^{\frac{1}{1-p}} k^{\frac{p}{p-1}} \cdot \log^5 N\right)$*.*

Notice that Corollary 5 does not guarantee the exact recovery of k-frequency superpositions in the discrete setting. Generally, the sparse Fourier representations produced by Algorithm 2 on discrete data will always contain interpolation errors. However, for $\delta = \Theta\left(N^{-1}\right)$, we can still consider smooth data $\mathbf{A}$ to be Fourier $(c, \Theta(\log N))$-compressible and so achieve an accurate $\tilde{O}(k^2)$-time DFT algorithm for large N.

7 A Preliminary Empirical Evaluation

In this section we provide a preliminary empirical evaluation of Algorithm 2. More specifically, we empirically verify and test the deterministic sampling and runtime requirements for Algorithm 2 stated in Theorem 3. In the process we will also discuss implementation details which lead to better performance in practice. We will begin by considering the sampling requirements of Algorithm 2. For ease of discussion, throughout the remainder of this section we will focus on recovering signals, $f : [0, 2\pi] \to \mathbb{C}$, containing *exactly* $B = k$ nonzero frequencies in $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$.

7.1 Sampling Performance

The number of samples utilized by Algorithm 2 is entirely determined by the chosen $p_0, \ldots, p_m$ and $\mathcal{S}_j$-primes (see Equation 26 and Section 3). More specifically, the number of times Algorithm 2 evaluates f equals

$$\sum_{j=0}^{K} \sum_{l=0}^{m} p_l \cdot p_{q+j} = \left(\sum_{l=0}^{m} p_l\right) \cdot \left(\sum_{j=0}^{K} p_{q+j}\right). \tag{29}$$

In Section 5 we chose our $p_0, \ldots, p_m$ and $\mathcal{S}_j$-primes in order to both (i) comply with Lemma 1, and (ii) allow us to express the sampling and runtime complexities of Algorithm 2 in terms of N and B. However, in practice we can substantially decrease the sampling usage of Algorithm 2 by changing the relatively prime values it uses (e.g., by replacing its $p_0, \ldots, p_m$ primes and $\mathcal{S}_j$-primes with other values as described below).

In order to recover an exactly Fourier B-sparse signal, f, with bandwidth N using the fewest possible samples via the methods herein we must choose new 'p_l' and '$\mathcal{S}_j$' integers (not necessarily prime) that minimize Equation 29 subject to the CRT-derived reconstruction requirements discussed in Sections 1.1 and 3. To emphasize that we will no longer require Algorithm 2 to use primes, let us replace its $K+1$ $\mathcal{S}_j$-primes with $K+1$ s_j-integers,

$$s_0 \le s_1 \le \cdots \le s_j \le \cdots \le s_K.$$

Furthermore, let each s_j-integer have m_j associated $r_{j,l}$ integers, $1 \le l \le m_j$. These $r_{j,l}$ integers will play the role formally played by our $p_1, \ldots, p_m$ primes. Optimizing the sample usage of Algorithm 2 is now equivalent to minimizing

$$\sum_{j=0}^{K} \sum_{l=1}^{m_j} (r_{j,l} + 1) \cdot s_j = \sum_{j=0}^{K} s_j \cdot \left(1 + \sum_{l=1}^{m_j} r_{j,l}\right) \tag{30}$$

subject to the following constraints:

1. $\frac{K}{2B} \in \mathbb{N}$,
2. $\prod_{j=0}^{\frac{K}{2B}} s_j \geq N$,
3. $s_0, \cdots, s_K$ are pairwise relatively prime,
4. $\prod_{l=1}^{m_j} r_{j,l} \geq \frac{N}{s_j}$ for each $j \in [0, K] \cap \mathbb{Z}$, and
5. $s_j, r_{j,1}, \ldots, r_{j,m_j}$ are pairwise relatively prime for each $j \in [0, K] \cap \mathbb{Z}$.

Unfortunately, it is unclear how to solve this optimization problem in a computationally efficient manner. Naively, it appears as if the pairwise relatively prime properties of $\binom{N^{\Omega(1)}}{B^{\Omega(1)}}$ sets of possible s_j-integers have to be checked during the solution process.

In Section 5 we approximated the optimal solution to the sampling problem for Algorithm 2 (i.e., Equation 30) by setting

1. m to be the smallest integer such that $\prod_{l=0}^{m} p_l \geq \frac{N}{B}$,
2. q so that $p_{q-1} \leq \max(B, p_m) < p_q$,
3. $K = 2B \left\lfloor \log_{p_q} N \right\rfloor$,
4. $s_j = p_{q+j}$ for all $j \in [0, K] \cap \mathbb{Z}$,
5. $m_j = m$ for all $j \in [0, K] \cap \mathbb{Z}$,
6. $r_{j,l} = p_l$ for all $j \in [0, K] \cap \mathbb{Z}$, and $l \in [1, m] \cap \mathbb{Z}$.

Although, as mentioned above, using this approximate solution allows us to obtain sampling bounds in terms of B and N, it also leads to suboptimal sampling by Algorithm 2 in practice. See Algorithm 3 for a simple procedure which quickly chooses s_j and $r_{j,l}$ integers by greedily minimizing portions of Equation 30. Preliminary experiments indicate that Algorithm 3 tends to minimize Equation 30 better than the techniques utilized in Section 5.

Algorithm 3 attempts to decrease the number of samples used by Algorithm 2 in two ways. First, note that using a larger $s_0 = p_q$ for Algorithm 2 sometimes reduces the total sampling requirements by decreasing the number of $\mathcal{S}_j$-primes, $K \geq 2B \left\lfloor \log_{p_q} N \right\rfloor$, required by Lemma 1. Hence, it is technically possible to increase p_q and, as a consequence, end up requiring a subset of the previously required $\mathcal{S}_j$-primes. Based on this observation, Algorithm 3 attempts to increase the first utilized prime, p_q, to a new value, $p_{q_{min}}$, which minimizes the resulting sum of $K \geq 2B \left\lfloor \log_{p_{q_{min}}} N \right\rfloor$ primes.

Second, Algorithm 3 uses powers of p_l primes to help decrease Algorithm 2's sample usage. Instead of performing DFTs of size $p_0 \cdot p_{q+j}, \ldots, p_m \cdot p_{q+j}$ for each prime p_{q+j}, Algorithm 3 allows Algorithm 2 to compute DFTs of size

$$p_0 \cdot p_{q_{\min}+j}, p_1^{\alpha_{j,1}} \cdot p_{q_{\min}+j}, \ldots, p_l^{\alpha_{j,l}} \cdot p_{q_{\min}+j}, \ldots, p_{m_j}^{\alpha_{j,m_j}} \cdot p_{q_{\min}+j}$$

for each prime $p_{q_{\min}+j}$. Here m_j is chosen so that

$$\prod_{l=1}^{m_j} p_l^{\alpha_{j,l}} \geq \frac{N}{p_{q_{\min}+j}}$$

for each $p_{q_{\min}+j}$. As we shall see next, allowing Algorithm 2 the freedom to use powers of p_l primes can help to significantly reduce sample usage.

Consider an $N = 60,000$ bandwidth signal, f, containing exactly $B = 5$ non-zero frequencies. Algorithm 2, as formulated in Section 5, would set m equal to 6 and use $2, 3, 5, 7, 11,$ and 13 as its p_l-primes (i.e., its $r_{j,l}$-integers for all j). This would make q equal to 7 and p_q equal to 17. Therefore, $K = 2 \cdot 5 \cdot \lfloor \log_{17} 60,000 \rfloor = 30$ which would set

Algorithm 3 CHOOSE s_j AND $r_{j,l}$ VALUES

```
1:  Input: integers $B, N$ with $B < N$
2:  Output: $s_j$ and $r_{j,l}$ integers
3:  $m \leftarrow \min\left\{\beta \mid \prod_{l=0}^{\beta} p_l \geq \frac{N}{B}\right\}$
4:  $q_{\min} \leftarrow \arg\min_{\tilde{q}:\ \max\{B,m\}<p_{\tilde{q}}<N} \sum_{j=0}^{2B\left\lfloor \log_{p_{\tilde{q}}} N \right\rfloor} p_{\tilde{q}+j}$
5:  $K \leftarrow 2B\left\lfloor \log_{p_{q_{\min}}} N \right\rfloor$
6:  for $j$ from 0 to $K$ do
7:      Initialize $\alpha_{j,0} \leftarrow 1$, $\alpha_{j,l} \leftarrow 0$ for $1 \leq l \leq m$
8:      $s_j \leftarrow p_{q_{\min}+j}$
9:      while $\prod_{l=0}^{m} p_l^{\alpha_{j,l}} < \frac{N}{s_j}$ do
10:         $l_{\min} \leftarrow \arg\min_{l:\ 1\leq l\leq m} p_l^{\alpha_{j,l}}(p_l - 1) + \chi_{\{0\}}(\alpha_{j,l})$
11:         $\alpha_{j,l_{\min}} \leftarrow \alpha_{j,l_{\min}} + 1$
12:     end while
13:     $l \leftarrow 0$
14:     while $\alpha_{j,l} > 0$ do
15:         $r_{j,l} \leftarrow p_l^{\alpha_{j,l}}$
16:         $l \leftarrow l + 1$
17:     end while
18:     $m_j \leftarrow l - 1$
19: end for
```

$s_0 = 17,\ s_1 = 19,\ \ldots,\ s_{30} = 157$. The resulting total number of samples would therefore be $42 \cdot (17 + 19 + \cdots + 157) = 42 \cdot 2543 = 106{,}806$.

However, if we employ Algorithm 3 we will set $q_{\min}$ to be 13 and have $p_{q_{\min}}$ equal to 41. Thus, Algorithm 3 will use $K = 2 \cdot 5 \cdot \lfloor \log_{41} 60{,}000 \rfloor = 20$ and set $s_0 = 41,\ \ldots,\ s_{20} = 137$. In this case it turns out that the $r_{0,l}$-integers will be $8, 9, 5$, and 7 for all the s_j-integers. The resulting total number of samples will therefore be $30 \cdot (41 + 43 + \cdots + 137) = 30 \cdot 1791 = 53{,}730$. All told, we can see that for this example Algorithm 3 decreases the sampling requirements of Algorithm 2 to about half of what they are when Section 5 methods are employed.

Finally, note that one must be able to obtain lists of prime numbers in order to employ Algorithm 3. Sieving algorithms for generating the first n primes exist (see [47]), and have been widely implemented (e.g., MATLAB's PRIMES function). In addition, previously computed lists of primes can be found online (e.g., the first 50 million primes can be downloaded at [1]).

Given that a standard FFT can determine the Fourier transform of an N-bandwidth signal f by taking N samples from f, it is important for us to determine when Algorithm 2 enables us to utilize less than N samples to recover $\widehat{f}$. Figure 1 addresses this issue by plotting, for each bandwidth value N, the maximum number of nonzero frequencies f may contain while still allowing Algorithm 2 to determine $\widehat{f}$ using less than N f-samples. The number of samples required by Algorithm 1 is also included for reference (see Theorem 2). In both cases the s_j-integers (and $r_{j,l}$-integers in the case of Algorithm 2) were generated using Algorithm 3. All bandwidth values, N, are powers of two.

Looking at Figure 1 we can see that Algorithm 1 can use a sublinear number of samples to recover signals containing roughly an order of magnitude more frequencies than can be recovered by Algorithm 2 using sublinear sampling. For example, Algorithm 2 can recover signals containing at most 9 nonzero frequencies at bandwidth 2^{18} using fewer than 2^{18} samples, whereas Algorithm 1 can recover signals containing over 100 nonzero frequencies. Generally, Algorithm 2 pays for its better runtime complexity by using more signal samples.

Figure 2 contains the number of function evaluations used by Algorithm 2 to recover signals with B nonzero frequencies at five different bandwidth values. As before, the s_j and $r_{j,l}$ integers utilized by Algorithm 2 were generated by Algorithm 3. The vertical axis of Figure 2 is in terms of bandwidth-fraction sampled $\left(\text{i.e., } \frac{\text{samples}}{\text{bandwidth } N} \text{ for each curve}\right)$. Thus, for example, we can see that Algorithm 2 can recover any 10-frequency function with bandwidth 2^{20} by sampling

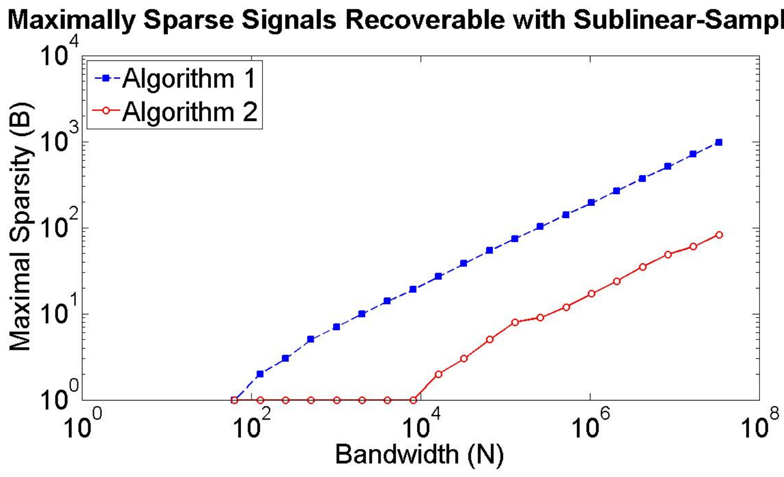

Figure 1: Maximal Fourier Sparsity a Signal May Have and Be Recovered Deterministically Via Sublinear-Sampling

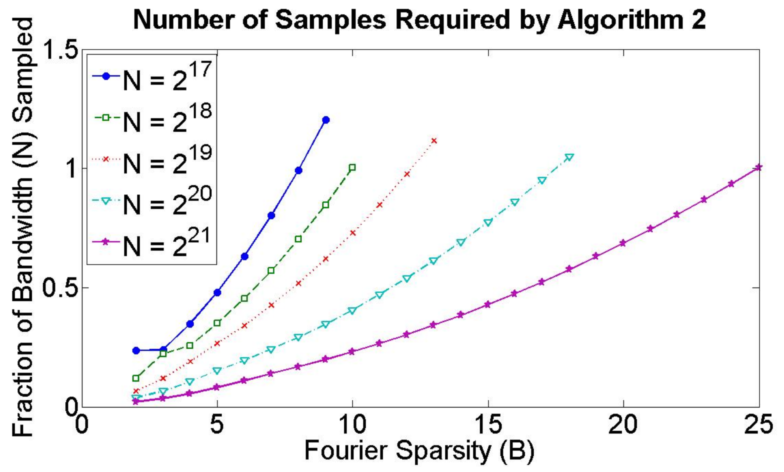

Figure 2: Number of Samples Required by Algorithm 2 to Deterministically Reconstruct Signals

it less than 2^{19} times (i.e., by using less than half the samples a full FFT requires).

We conclude this section with an empirical verification of the sampling complexity for Algorithm 2 stated in Theorem 3. When N is fixed, we expect the number of samples produced by both the methods of Section 5 and Algorithm 3 to scale quadratically in B. See Figure 3 for an empirical verification of the quadratic behavior of Algorithm 2 when $N = 2^{22}$. Figure 3 graphs both the number of samples used by Algorithm 2 and the best fit quadratic function for number of samples used by Algorithm 2 for various Fourier sparsity levels B. The absolute error between the sample usage of Algorithm 2 and its best fit quadratic function is also graphed. As we can see from the figure, the best fit quadratic agrees quite well with Algorithm 2's sampling data for larger B. Indeed, the relative error between the number of samples used by Algorithm 2 and its best fit quadratic function was less than 0.01 for all $B \geq 10$.

7.2 Runtime Performance

Both Algorithm 1 and 2 were implemented in C in order to empirically evaluate their runtime characteristics. More specifically, each line 7/8 Algorithm 1/2 discrete Fourier transform was performed using FFTW 3.2 [23] with an

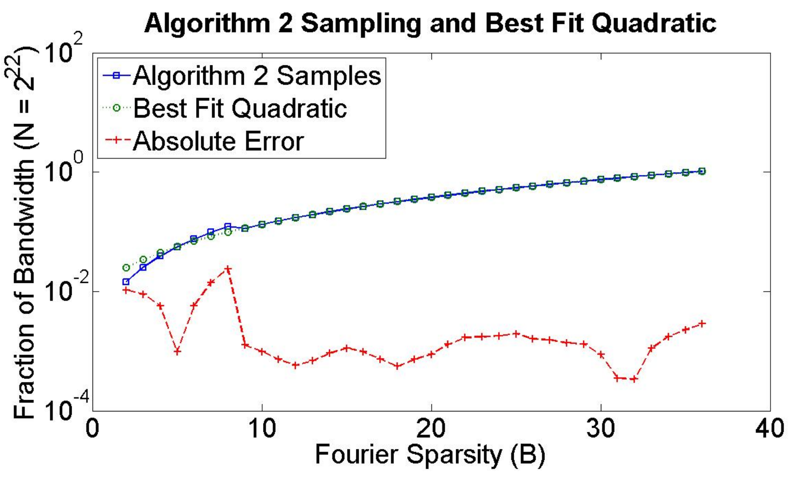

Figure 3: Quadratic Behavior of the Number of Samples Used by Algorithm 2 with Fixed $N = 2^{22}$

FFTW_MEASURE plan. All line 10 and 11 (as well as line 26 and 27) medians, together with all sorting steps in both Algorithm 1 and 2, were implemented with C code for Quick Sort (see [10]). Algorithm 3 was also coded in C and used to find the s_j (and $r_{j,l}$) integers for Algorithm 1 (and 2). Finally, the reconstruction of each frequency from its remainders in line 19 of Algorithm 2 was implemented using standard CRT-related reconstruction techniques based on the Euclidean algorithm (see [47]).

All experiments were run on a QuadCore2 2.4 Ghz Ubuntu Linux machine with 3 GB of RAM. We used FFTW 3.2 with an FFTW_MEASURE plan as our FFT for runtime comparisons in Figures 4 and 5. All bandwidth values (i.e., array lengths) used for generating our graphs were powers of two. FFTW 3.2 is a highly optimized FFT implementation which adapts itself to the computer on which it is executed. Under these conditions (i.e., using an FFTW_MEASURE plan with the known bandwidth, N, increased to the nearest power of two) one can expect FFTW 3.2 to run near its fastest capabilities for each signal on the given machine.

All N-bandwidth B nonzero frequency signals used for tests below where constructed as follows: First, B frequencies were selected uniformly at random from $\left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right]$. Next, each randomly selected frequency was given a uniformly random phase. Their coefficient magnitudes were left as 1. Despite the fact that both Algorithm 1 and 2 are guaranteed to deterministically recover any such signal, every data point in both Figure 4 and 5 is the result of 1000 runs on randomly generated signals. During all runs the errors of both algorithms were monitored. Their precisions were within an order of magnitude of FFTW's for all recovered frequency coefficients in all tests reported on below.

Figure 4 contains graphs of both Algorithm 1 and 2's runtimes (averaged over 1000 runs per data point) for 1024-bandwidth signals containing various numbers of nonzero frequencies. FFTW's runtime is included for reference. As we can see, Algorithm 2 is indeed faster than Algorithm 1 for all sparsity levels despite the modest bandwidth value. Both Algorithms required less than 5 ms for all recorded runs. However, FFTW 3 is by far the fastest DFT method for smaller bandwidth values.

Figure 5 plots the runtimes of both Algorithm 1 and 2 for signals containing 8 nonzero frequencies hidden in various bandwidths. Looking at Figure 5 we can see that Algorithm 2 is faster than Algorithm 1 for all bandwidth values greater than 128. Likewise, Algorithm 2 is faster than FFTW for all bandwidth values greater than 2^{18}. More generally, Algorithm 2 will be faster than FFTW for all highly-sparse wideband signals.

It is worth mentioning that Algorithm 2 appears to be naturally suited to parallel implementation. Indeed, all the discrete Fourier transforms computed in line 8, sorts performed in line 12, and line 19 frequency reconstructions can be performed in parallel. In addition, the medians in lines 26 and 27 can also be done in parallel once all frequently occurring frequencies have been identified. Once the function samples have been provided, the only real barrier to a straightforward parallel implementation appears to be the compilation of recovered frequencies in line 22. It would be interesting to pursue a parallel implementation as part of a more thorough empirical evaluation of the sparse Fourier

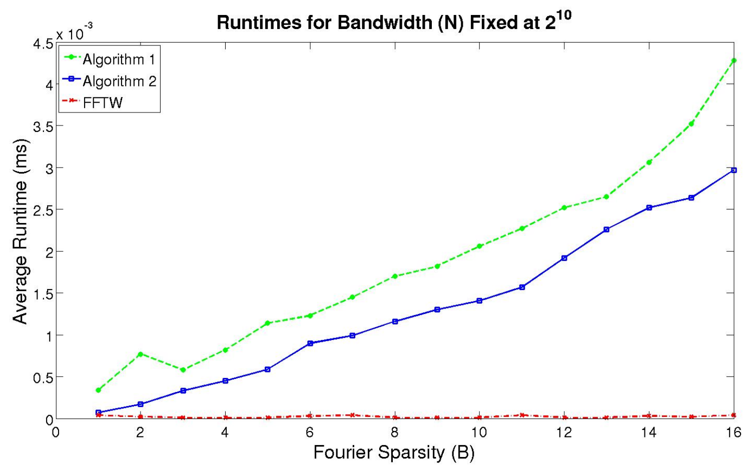

Figure 4: Runtime Comparison at Fixed Bandwidth, $N = 2^{10}$

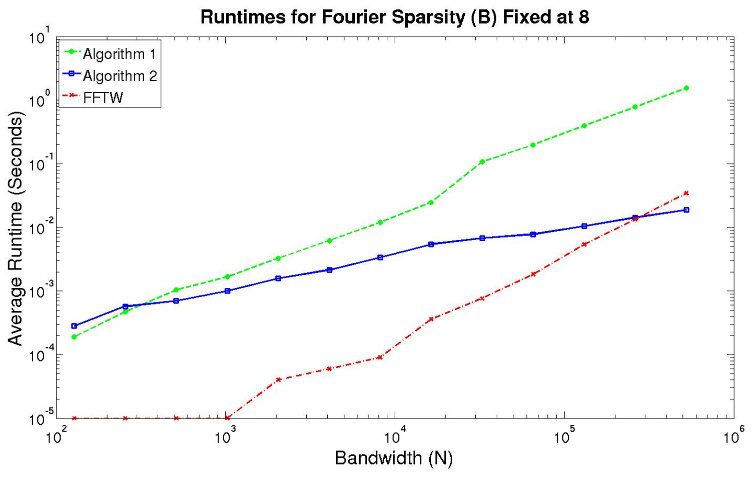

Figure 5: Runtime Comparison at Fixed Fourier Sparsity, $B = 8$

methods presented herein.

Finally, the s_j and $r_{j,l}$ integers selected for Algorithm 2 also impact its runtime. It could prove fruitful to explore alternative methods to Algorithm 3 for minimizing Equation 30. In particular, it would be interesting to determine theoretically whether minimizing Equation 30 subject to the given constraints is truly as difficult as it appears at first glance.

8 Conclusion

In this paper the first known deterministic Fourier algorithm with both sublinear-time sampling and runtime complexity was developed. Hence, we have established the first deterministic algorithm which can exactly reconstruct a k-frequency superposition using time polynomial in the superposition's *information content*. When viewed from this perspective the following open problem presents itself.

Open Problem 1. *Construct (or show the impossibility of constructing) a deterministic Fourier algorithm guaranteed* *to exactly recover k-frequency superpositions in* $k \cdot \log^{O(1)} N$ *time.*

The status of current methods with respect to Problem 1 is as follows: Gilbert, Muthukrishnan, and Strauss' randomized Fourier algorithm [25] achieves a near optimal runtime, but is neither deterministic nor exact. Similarly, our Section 5 Monte Carlo algorithm achieves exact reconstruction and a near optimal runtime, but is not deterministic. Linear programming [15, 5] and OMP-based [46] methods achieve universal sampling sets of acceptable size [49, 16], but both the verification of the sampling sets universal properties and the associated reconstruction procedures are computationally taxing. Finally, Indyk's fast deterministic CS procedure [29] obtains a promising reconstruction runtime, but does not allow fast Fourier measurement acquisition.

In terms of applications, there are two compelling motivations for developing fast sparse Fourier transform methods along the lines of [24, 25] and Algorithm 2: runtime and sample usage. In numerical applications such as [13] where runtime is the dominant concern we must assume that our input function f exhibits some multiscale behavior. If $\hat{f}$ contains no unpredictably energetic and large (relative to the number of desired Fourier coefficients) frequencies then it is more computationally efficient to simply use standard FFT/NUFFT methods [9, 37, 2, 18, 21]. In other applications [36, 34, 38, 39] where sampling costs are of greater concern than reconstruction runtime, even mild oversampling for the sake of faster reconstruction may be unacceptable. In such cases the runtime/sampling tradeoff must be carefully weighed.

Acknowledgments

I would like to thank the anonymous reviewers and Tsvetanka Sendova for suggesting many helpful improvements to the preliminary drafts of this paper.

References

[1] The Prime Pages. http://primes.utm.edu/.

[2] C. Anderson and M. D. Dahleh. Rapid computation of the discrete Fourier transform. *SIAM J. Sci. Comput.*, 17:913–919, 1996.

[3] L. I. Bluestein. A Linear Filtering Approach to the Computation of Discrete Fourier Transform. *IEEE Transactions on Audio and Electroacoustics*, 18:451–455, 1970.

[4] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, Inc., 2001.

[5] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theory*, 52:489–509, 2006.

[6] E. Candes, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[7] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Brooks/Cole Publishing Company, 1992.

[8] A. Cohen, W. Dahmen, and R. DeVore. Compressed Sensing and Best k-term Approximation. *Journal of the American Mathematical Society*, 22(1):211–231, January 2008.

[9] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19:297–301, 1965.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[11] G. Cormode and S. Muthukrishnan. Combinatorial Algorithms for Compressed Sensing. *Technical Report DIMACS TR 2005-40*, 2005.

[12] G. Cormode and S. Muthukrishnan. Combinatorial Algorithms for Compressed Sensing. *Conference on Information Sciences and Systems*, March 2006.

[13] I. Daubechies, O. Runborg, and J. Zou. A sparse spectral method for homogenization multiscale problems. *Multiscale Model. Sim.*, 2007.

[14] R. A. DeVore. Deterministic constructions of compressed sensing matrices. *Journal of Complexity*, 23, August 2007.

[15] D. Donoho. Compressed Sensing. *IEEE Trans. on Information Theory*, 52:1289–1306, 2006.

[16] D. L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. In *40th Annual Conference on Information Sciences and Systems (CISS)*, 2006.

[17] D. Z. Du and F. K. Hwang. *Combinatorial Group Testing and Its Applications*. World Scientific, 1993.

[18] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14:1368–1383, 1993.

[19] D. Eppstein, M. T. Goodrich, and D. S. Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *http://arxiv.org/abs/cs.DS/0505048*, May 2005.

[20] L. Y. Erich Kaltofen. Improved sparse multivariate polynomial interpolation algorithms. *International Symposium on Symbolic and Algebraic Computation*, 1988.

[21] J. A. Fessler and B. P. Sutton. Nonuniform Fast fourier transforms using min-max interpolation. *IEEE Trans. Signal Proc.*, 51:560–574, 2003.

[22] G. B. Folland. *Fourier Analysis and Its Applications*. Brooks/Cole Publishing Company, 1992.

[23] M. Frigo and S. Johnson. The design and implementation of fftw3. *Proceedings of IEEE 93 (2)*, pages 216–231, 2005.

[24] A. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier estimation via sampling. *ACM STOC*, pages 152–161, 2002.

[25] A. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse Fourier representations. *Proceedings of SPIE Wavelets XI*, 2005.

[26] A. C. Gilbert and M. J. Strauss. Group testing in statistical signal recovery. *preprint*, 2006.

[27] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *preprint*, 2006.

[28] P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 697–704, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[29] P. Indyk. Explicit constructions for compressed sensing of sparse signals. In *Proc. of ACM-SIAM symposium on Discrete algorithms (SODA'08)*, 2008.

[30] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *Proc. of ACM-SIAM symposium on Discrete algorithms (SODA'08)*, 2008.

[31] M. A. Iwen, A. C. Gilbert, and M. J. Strauss. Empirical evaluation of a sub-linear time sparse DFT algorithm. *Communications in Mathematical Sciences*, 5(4), 2007.

[32] M. A. Iwen and C. V. Spencer. Improved bounds for a deterministic sublinear-time sparse fourier algorithm. In *Conference on Information Sciences and Systems (CISS)*, 2008.

[33] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. China Machine Press, 2003.

[34] S. Kirolos, J. Laska, M. Wakin, M. Duarte, D. Baron, T. Ragheb, Y. Massoud, and R. Baraniuk. Analog-to-information conversion via random demodulation. *Proc. IEEE Dallas Circuits and Systems Conference*, 2006.

[35] S. Kunis and H. Rauhut. Random Sampling of Sparse Trigonometric Polynomials II - Orthogonal Matching Pursuit versus Basis Pursuit. *Foundations of Computational Mathematics*, 8(6):737–763, 2008.

[36] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. Gilbert, M. Iwen, and M. Strauss. Random sampling for analog-to-information conversion of wideband signals. *Proc. IEEE Dallas Circuits and Systems Conference*, 2006.

[37] J.-Y. Lee and L. Greengard. The type 3 nonuniform FFT and its applications. *J Comput. Phys.*, 206:1–5, 2005.

[38] M. Lustig, D. Donoho, and J. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, Dec. 2007.

[39] R. Maleh, A. C. Gilbert, and M. J. Strauss. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Int. Conf. on Image Processing*, 2007.

[40] S. Mallet. *A Wavelet Tour of Signal Processing*. China Machine Press, 2003.

[41] Y. Mansour. Learning boolean functions via the fourier transform. *Theoretical Advances in Neural Computation and Learning*, pages 391–424, 1994.

[42] Y. Mansour. Randomized approxmation and interpolation of sparse polynomials. *SIAM Journal on Computing*, 24:2, 1995.

[43] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[44] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1, 2005.

[45] S. Muthukrishnan. Some Algorithmic Problems and Results in Compressed Sensing. *Allerton Conference*, 2006.

[46] D. Needell and R. Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9:317–334, 2009.

[47] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An Introduction to The Theory of Numbers*. John Wiley & Sons, Inc., 1991.

[48] L. Rabiner, R. Schafer, and C. Rader. The Chirp z-Transform Algorithm. *IEEE Transactions on Audio and Electroacoustics*, AU-17(2):86–92, June 1969.

[49] M. Rudelson and R. Vershynin. Sparse reconstruction by convex relaxation: Fourier and gaussian measurements. In *40th Annual Conference on Information Sciences and Systems (CISS)*, 2006.

[50] J. Tropp and A. Gilbert. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Trans. Info. Theory*, 53(12):4655–4666, Dec. 2007.