Noname manuscript No.
(will be inserted by the editor)

Improved Sparse Fourier Approximation Results

Faster Implementations and Stronger Guarantees

Ben Segal · M. A. Iwen

Received: date / Accepted: date

Abstract We study the problem of quickly estimating the best k-term Fourier representation for a given periodic function $f : [0, 2\pi] \to \mathbb{C}$. Solving this problem requires the identification of k of the largest magnitude Fourier series coefficients of f in worst case $k^2 \cdot \log^{O(1)} N$ time. Randomized sublinear-time Monte Carlo algorithms, which have a small probability of failing to output accurate answers for each input signal, have been developed for solving this problem [17,18]. These methods were implemented as the Ann Arbor Fast Fourier Transform (AAFFT) and empirically evaluated in [23]. In this paper we present a new implementation, called the Gopher Fast Fourier Transform (GFFT), of more recently developed sparse Fourier transform techniques [21, 22]. Experiments indicate that GFFT is faster than AAFFT.

In addition to our empirical evaluation, we also consider the existence of sublinear-time Fourier approximation methods with deterministic approximation guarantees for functions whose sequences of Fourier series coefficents are compressible. In particular, we prove the existence of sublinear-time Las Vegas Fourier Transforms which improve on the recent deterministic Fourier approximation results of [21,22] for Fourier compressible functions by guaranteeing accurate answers while using an asymptotically near-optimal number of function evaluations.

Keywords Fourier Analysis · Sparse Approximation · Fast Fourier Transforms · Randomized Algorithms

Research supported in part by ONR N00014-07-1-0625 and NSF DMS-0847388.

Ben Segal
Ben died unexpectedly at the age of 23 in January, 2012, after graduating from the University of Minnesota the previous May with degrees in both mathematics and economics. He was a talented, curious, and enthusiastic student who loved mathematics for its innate beauty.

M. A. Iwen
Duke University Mathematics Department
Durham, NC 27708
E-mail: markiwen@math.duke.edu

1 Introduction

In many applications only a few largest magnitude terms in the Fourier series of a given periodic function, $f : [0, 2\pi] \to \mathbb{C}$, are of interest. In such situations the Fast Fourier Transform (FFT) [11], which approximates the function by a trigonometric polynomial of degree N in $O(N \log N)$-time, can be computationally wasteful. This is especially true when f has a small number of high-frequency Fourier coefficients with relatively large magnitudes (i.e., because N must be chosen to be much larger than the small number of energetic frequencies in such cases). Functions of this type arise naturally in application areas such as MR imaging [28] and wideband analog-to-digital conversion [27, 24]. In these settings the primary difficulty is acquiring enough function samples on which to reliably perform an (inverse) Fourier transform in the first place. Over the past several years compressed sensing methods (e.g., see [14, 6, 8]) have made astonishing progress toward reducing the number of function evaluations required in order to approximate such Fourier sparse signals. However, despite their small sampling requirements, all aforementioned compressed sensing methods have computational complexities which greatly exceed that of a standard FFT. Hence, these methods are not well suited for large scale problems (i.e., N large) when runtime is a dominant consideration.

This paper is primarily concerned with the computational complexity of approximating high-degree trigonometric polynomials which have a small number of coefficients with relatively large magnitudes. In particular, we will focus on sublinear-time Fourier approximation techniques which are capable of approximating such trigonometric polynomials more quickly than a standard FFT. Fast algorithms of this type were first developed by Mansour et al. for function learning problems [26, 30, 31]. Similar sublinear-time Fourier algorithms based on discrepancy methods [9] were then developed by Akavia et al. for cryptographic applications [2, 1]. Later, Gilbert et al. reduced the runtime complexity of such fast sparse Fourier methods to within logarithmic factors of the optimal runtime dependence [17, 18]. Their Fourier algorithm [18] remained the only sublinear-time Fourier algorithm with both instance optimal error bounds and a near-optimal runtime complexity until the recent development of another such algorithm in [22]. In Section 3 of this paper an implementation of the sparse Fourier approximation techniques from [22], the Gopher Fast Fourier Transform (GFFT), is introduced. GFFT is then empirically evaluated against both an existing implementation of [18], the Ann Arbor Fast Fourier Transform (AAFFT) [23], and a highly optimized standard Fast Fourier Transform implementation, FFTW3 [16]. This empirical evaluation demonstrates that a variant of GFFT is generally faster than AAFFT. Furthermore, it is demonstrated that both GFFT and AAFFT can reliably recover high-degree trigonometric polynomials with a small number of nonzero terms more quickly than standard FFT methods (i.e., faster than FFTW3).

Both AAFFT and the fastest GFFT variant evaluated in Section 3 below are based on randomized algorithms which have small probabilities of failing to approximate a given trigonometric polynomial to within a given tol-

erance. Although these methods are capable of approximating Fourier sparse signals faster than standard FFT methods, their randomized recovery guarantees can greatly complicate numerical methods in which they are utilized as subroutines. For example, when utilized as part of a spectral method [5] for solving multiscale homogenization problems as proposed in [13], such randomized Fourier methods must be applied repeatedly during each iteration of a time-stepping scheme. In this setting the randomized approximation guarantees of these Fourier algorithms ultimately force them to collectively use a number of operations that scales superlinearly (e.g., quadratically) in the number of required time steps in order to numerically solve the problem with guaranteed accuracy with high probability. Fast Fourier methods with approximation guarantees allow one to avoid these superlinear time stepping costs (i.e., because there is no probability of failure in each iteration which must be accounted for). Similar considerations have also driven the development of deterministic fast Fourier approximation algorithms in other application areas (e.g., see [1]).

Let $f : [0, 2\pi] \to \mathbb{C}$ be a trigonometric polynomial of degree N. Denote the Fourier series coefficients of f by $\hat{f}$. We will consider $\hat{f}$ to be an array of $2N+1$ complex numbers (i.e., all but $2N + 1$ Fourier series coefficients will be zero). Given samples from f we are interested in locating (and then approximating) $k \ll N$ of the largest magnitude entries of $\hat{f}$ as quickly as possible. Hence, we seek Fourier algorithms with runtime and sampling complexities that scale sublinearly in N, the degree of our trigonometric polynomial f. Any Fourier algorithm of this type will output a k-element list of $\hat{f}$'s largest magnitude values. We will refer to any such list as a *sparse Fourier representation*.

In addition to empirically evaluating GFFT against AAFFT and FFTW3 in Section 3, we also consider fast Fourier approximation algorithms which are guaranteed to produce near-optimal sparse Fourier representations for any given trigonometric polynomial in Section 4 (i.e., we consider Fourier approximation algorithms having no probability of failure). Randomized Fourier approximation algorithms which can produce a near-optimal k-term sparse Fourier representation for any trigonometric polynomial of degree N with high probability in $O(k \cdot \text{polylog}(N))$-time exist (e.g., see Theorem 1 below). It remains an open problem to determine whether or not a Fourier approximation algorithm exists which is guaranteed to always produce a near-optimal sparse Fourier representation for any trigonometric polynomial in $O(k \cdot \text{polylog}(N))$-time. The best result to date guarantees that a near-optimal k-term sparse Fourier representation can be found for any trigonometric polynomial of degree N in $O(k^2 \cdot \log^4 N)$-time after collecting $O(k^2 \cdot \log^4 N)$ evaluations of the polynomial (see Theorem 2 below). However, fast compressed sensing methods which utilize less constrained linear measurements can do slightly better. Gilbert et al. have shown the existence of algorithms which are guaranteed to produce a near-optimal k-term sparse approximation of any given vector, $\mathbf{x} \in \mathbb{R}^N$, in $O(k^2 \cdot \text{polylog}(N))$-time using $O(k \cdot \text{polylog}(N))$ linear measurements (see [20]). Although these less constrained compressed sensing results

do not translate into Fourier approximation results of the type we consider herein, they do indicate that it might be possible to prove a similar result in the Fourier setting. In Section 4 we show that this is indeed the case by proving the existence of Las Vegas Fourier approximation algorithms (e.g., Algorithm 1) which are guaranteed to always produce near optimal k-term sparse Fourier approximations for any trigonometric polynomial of degree N in expected $O(k^2 \cdot \log^4(N))$-time using at most $O(k \cdot \log^4(N))$ evaluations of the polynomial. This maintains the runtime complexity of Theorem 2 (in expectation) while simultaneously reducing its worst case sampling complexity.

Finally, we note that the methods considered in this paper for approximating trigonometric polynomials are also applicable to the approximation of related polynomials. For example, the algorithms considered herein also apply to Chebyshev polynomials after a change of variables (see [5] for a detailed discussion concerning the relationships between these types of polynomials).

The remainder of this paper is organized as follows: In Section 2 we introduce the definitions, terminology, and theorems that are used as building blocks for subsequent results. In Section 3 we introduce a new implementation of the sparse Fourier approximation techniques from [22], the Gopher Fast Fourier Transform (GFFT). To demonstrate its capabilities, we empirically evaluate GFFT against both the Ann Arbor Fast Fourier Transform (AAFFT) [23], and a highly optimized standard Fast Fourier Transform implementation, FFTW3 [16]. Next, in Section 4, we discuss a simple strategy for improving the sampling requirements of existing sparse Fourier approximating results [21,22] for Fourier compressible signals. This strategy ultimately leads to the development of the Las Vegas Fourier algorithms mentioned above (e.g., Algorithm 1). Finally, we conclude with a brief summary of our results and observations in Section 5.

2 Preliminaries

We are interested in trigonometric polynomials, $f : [0, 2\pi] \to \mathbb{C}$, which are approximately Fourier sparse. Throughout the remainder of this section we will denote the degree of f by N. Hence, we are justified in considering the Fourier series coefficients of f, $\hat{f}$, to be a finite length vector in $\mathbb{C}^{2N+1}$. Because f is approximately Fourier sparse we assume only $k < 2N + 1$ entries of $\hat{f}$ contain values that are significant (or large) in magnitude. Given such a signal, fast Fourier approximation algorithms (e.g., [17,18,21,22,32]) produce output of the form $(\omega_1, C_1), \ldots, (\omega_k, C_k)$, where each $(\omega_m, C_m) \in (\mathbb{Z} \cap [-N, N]) \times \mathbb{C}$. We will refer to any such set of $k < 2N + 1$ tuples

$$\mathbf{R}^{\mathrm{s}} = \{(\omega_m, C_m) \text{ s.t. } m \in [1, k] \cap \mathbb{N}\}$$

as a *k-sparse representation*. Note that if we are given a sparse representation, $\mathbf{R}^{\mathrm{s}}$, we may consider it to be a vector (or signal) of length $2N+1$, $\mathbf{R} \in \mathbb{C}^{2N+1}$, defined by

$$\mathbf{R}_\omega = \begin{cases} C_\omega \text{ if } (\omega, C_\omega) \in \mathbf{R}^{\mathrm{s}} \\ 0 \quad \text{otherwise} \end{cases}$$

for all $\omega \in [-N, N] \cap \mathbb{Z}$.

Denote the discrete ℓ_q-norm of vector $\mathbf{a} \in \mathbb{C}^{2N+1}$ by

$$\|\mathbf{a}\|_q = \left(\sum_{j=-N}^{N} |a_j|^q \right)^{\frac{1}{q}}.$$

Furthermore, let $\boldsymbol{\omega} \in \mathbb{Z}^N$ be the first vector in lexicographical order whose entries sort a given trigonometric polynomial's Fourier series coefficients, $\hat{f} \in \mathbb{C}^{2N+1}$, by magnitude so that

$$|\hat{f}_{\omega_1}| \geq |\hat{f}_{\omega_2}| \geq \cdots \geq |\hat{f}_{\omega_N}|.$$

We define the optimal k-sparse representation for $\hat{f}$, $\mathbf{R}^{\mathrm{s}}_{\mathrm{opt}(k)}$, to be $(\omega_1, \hat{f}_{\omega_1}), \ldots, (\omega_k, \hat{f}_{\omega_k})$. All of the aforementioned sparse Fourier approximation algorithms attempt to recover k of the largest magnitude Fourier coefficients of $\hat{f}$ using as little time/samples from f as possible. Hence, the quality of their output sparse representations is measured with respect to the best possible k-sparse approximation errors, $\|\hat{f} - \mathbf{R}_{\mathrm{opt}(k)}\|_2$ and $\|\hat{f} - \mathbf{R}_{\mathrm{opt}(k)}\|_1$.

A variant of the following sparse Fourier approximation theorem is proven in [18].

Theorem 1 ***(Gilbert, Muthukrishnan, Strauss).*** *Let $f : [0, 2\pi] \to \mathbb{C}$ be a trigonometric polynomial of degree N. Fix precision parameters $\eta, \tau \in \mathbb{R}^+$ and probability parameter $\lambda \in (0, 1)$. There exists a randomized sampling algorithm which, when given sampling access to f, will output a k-sparse representation for $\hat{f}$, $\mathbf{R}^{\mathrm{s}}$, satisfying*

$$\|\hat{f} - \mathbf{R}\|_2 \leq \sqrt{1+\tau} \cdot \max\left\{\eta, \|\hat{f} - \mathbf{R}_{\mathrm{opt}(k)}\|_2\right\}$$

with probability at least $1 - \lambda$. Both the runtime and sampling complexities are

$$k \cdot \mathrm{polynomial}\left(\log\frac{1}{\lambda}, \log\frac{1}{\eta}, \log\|\hat{f}\|_2, \log N, \frac{1}{\tau}\right).$$

Although the randomized Fourier sampling algorithm referred to in Theorem 1 will fail to output accurate results with some probability for each trigonometric polynomial, both the probability of failure and the approximation precision can be made arbitrarily small in theory. Of course, there are tradeoffs: increasing either the precision or the probability of successful approximation will increase both the runtime and sampling requirements of the algorithm by logarithmic factors. The algorithm referred to by Theorem 1 has been implemented. The implementation, called AAFFT, was empirically evaluated in [23].

The following approximation result was recently proven (see [22]) for an improved variant of a sparse Fourier approximation algorithm proposed in [21].

Table 1 Fourier Algorithms and Implementations

Fourier Result	D/R	Runtime	Samples	Implementation
FFT [11]	D	$O(N \cdot \log N)$	N	FFTW3 [16]
Theorem 3 in [22]	D	$O(N \cdot k \cdot \log^2 N)$	$O(k^2 \cdot \log^2 N)$	GFFT-Det-Slow
Corollary 3 in [22]	R	$O\left(N \cdot \log N\right)$	$O\left(k \cdot \log^2 N\right)$	GFFT-Rand-Slow
Theorem 1	R	$k \cdot \log^{O(1)} N$	$k \cdot \log^{O(1)} N$	AAFFT [23]
Theorem 2	D	$O(k^2 \cdot \log^4 N)$	$O(k^2 \cdot \log^4 N)$	GFFT-Det-Fast
Corollary 4 in [22]	R	$O\left(k \cdot \log^4 N\right)$	$O\left(k \cdot \log^4 N\right)$	GFFT-Rand-Fast

Theorem 2 ***(Iwen).*** *Let $f : [0, 2\pi] \to \mathbb{C}$ be a trigonometric polynomial of degree N. Fix precision parameter $\epsilon \in (0, 1)$. There exists a deterministic sampling algorithm which, when given sampling access f, will output a $2k$-sparse representation for $\hat{f}$, $\mathbf{R}^s$, satisfying*

$$\left\|\hat{f} - \mathbf{R}\right\|_2 \leq \left\|\hat{f} - \mathbf{R}_{\text{opt}(k)}\right\|_2 + \frac{\epsilon \cdot \left\|\hat{f} - \mathbf{R}_{\text{opt}(k/\epsilon)}\right\|_1}{\sqrt{k}}. \tag{1}$$

Both the runtime and sampling complexities are

$$O\left(\frac{k^2 \cdot \log^4 N}{\epsilon^2}\right).$$

Unlike Theorem 1, Theorem 2 provides a deterministic approximation guarantee for all functions. However, the sampling and runtime complexities of Theorem 2 scale quadratically in the sparsity parameter, k, instead of linearly as achieved by Theorem 1. In Section 4 we will present a sublinear-time result which simultaneously achieves the sampling complexity of Theorem 1 together with uniform approximation guarantees along the lines of Theorem 2: That is, it acheives uniform approximation guarantees for all signals of a general class together with a worst-case sampling complexity that scales linearly in sparsity. For now, however, we will simply point out that randomized versions of Theorem 2, which achieve the approximation error bound stated in Equation 1 with arbitrarily high probability for each individual input signal as per Theorem 1, also exist. See [21,22] for details. We have implemented variants of both the aforementioned randomized and deterministic algorithms from [21, 22]. For the remainder of this paper they will be collectively referred to as GFFT.

2.1 Algorithms and Implementations

Table 1 summarizes the algorithms and implementations considered in Section 3 of this paper. The first column of Table 1 lists the Fourier approximation results empirically evaluated herein. The second column denotes whether each related Fourier result exhibits Deterministic (D) approximation guarantees for

all functions, or Randomized (R) approximate recovery with high probability per individual function. The third and fourth columns of Table 1 list the runtime and sampling complexities of the Fourier results, respectively. The listed runtime and sampling complexities assume that all precision parameters (e.g., τ, η, ϵ in Theorems 1 and 2) are fixed constants. For the randomized methods it is also assumed that the probability of successful approximation is at least $1 - 1/N^{O(1)}$ (e.g., that λ in Theorem 1 is $1/N^{O(1)}$). In the third row of Table 1 the stated $O(N \log N)$ runtime for Corollary 3 holds only when the number of samples it utilizes is $O(N)$ (i.e., only if k is $O(N/\log^2 N)$). Finally, the fifth column of Table 1 lists the implementation of each Fourier algorithm tested in Section 3 below.

2.1.1 FFTW3

FFTW3 [16] is a highly efficient implementation of the standard FFT algorithm [11]. It has been systematically optimized over the course of the last decade and remains one of the fastest freely available FFT implementations available today.

2.1.2 AAFFT

AAFFT [23] is based on the algorithms introduced in [17,18]. It is an iterative approximation algorithm which repeatedly performs three steps: (*i*) identification of frequencies whose Fourier coefficients are large in magnitude, (*ii*) accurate estimation of the Fourier coefficients of the frequencies identified in the first step, and (*iii*) subtraction of the contribution of the partial Fourier representation computed by the first two steps from the as-yet-unused function samples. Each repetition of the three steps above is guaranteed to gather a substantial fraction of the energy present in any given (approximately) Fourier sparse signal with high probability, effectively increasing the Fourier sparsity of the given input signal from one repetition to the next. The end result is that a small number of repetitions will gather (almost) all of the signal energy with high probability, thereby approximating the sparse Fourier transform of the given signal.

Consider, for example, a trigonometric polynomial having exactly 100 nonzero terms (i.e., Fourier coefficients). The first round of AAFFT will generally find and accurately estimate a large fraction of these terms (e.g., three fourths of them, or 75 terms in this case). The contributions of the discovered terms are then subtracted off of the remaining samples. This effectively reduces the number of nonzero terms in the trigonometric polynomial that generated the remaining samples, leaving about 25 terms in the current example. The next repetition of the three phases is now executed as before, but with the smaller effective sparsity of ≈ 25. All terms will be found and estimated after a few repetitions with high probability.

Step (*i*) of each AAFFT repetition, which identifies energetic frequencies in the Fourier spectrum of f, is generally the most involved of the three re-

peated steps mentioned above. It consists of several ingredients, including: random sampling designed to randomly permute the Fourier spectrum of the input signal, filtering to separate the permuted Fourier coefficients into different "frequency bins", and Monte Carlo integration techniques to estimate the energy in subsets of each of the aforementioned frequency bins. Roughly speaking, step (i) works by randomly binning the Fourier coefficients of f into a small number of bins, and then performing a modified binary search within each bin in order to find energetic frequencies that are isolated within each bin. The randomness is introduced into the Fourier spectrum of f by restricting the values at which one samples f to a randomly chosen arithmetic progression modulo 2π. This has the effect of randomly permuting the Fourier coefficients of f. The resulting "randomized" version of $\hat{f}$ is then binned via a filter (i.e., by convolving the randomly selected f-samples with a discretized filter function). Because f is approximately sparse in the frequency domain, each bin is likely to receive only one relatively large Fourier coefficient. Each such isolated Fourier coefficient is then identified via a modified binary search method. Simply put, samples from the permuted and filtered version of f are used to effectively estimate the energy in two different halves of each frequency bin via Monte Carlo integration techniques. The more energetic of the two sides is then selected and further subdivided into two smaller halves. This process is repeated on ever smaller halves of the Fourier bin under consideration until a single energetic frequency in that bin has been localized. The collection of frequencies discovered in each bin by this modified binary search is then sent on to step (ii) above.

Note that one must use a filter that is highly concentrated in time (i.e., that is "essentially zero" everywhere, except at a small number of time coordinates) in order to allow fast convolution calculations to be performed with the filter. This is important because these convolutions are used to compute samples from the binned and randomized Fourier spectrum of f. The Fourier transform of the filter must also have a special structure for the same reason. In particular, the filter's Fourier transform must be very small everywhere except on a $1/\Theta(k \log^c N)$-fraction of the N smallest integer frequencies. The filter is then shifted $\Theta(k \log^c N)$ times in order to create $\Theta(k \log^c N)$ approximate "pass regions", each of size $N/\Theta(k \log^c N)$, which tile the entire Fourier spectrum of f under consideration. These shifted filters collectively form the "frequency bins" mentioned above. More specifically, AAFFT uses a collection of shifted and dilated rectangular filters for frequency binning.

Roughly speaking, the second and third steps repeated by AAFFT are comparatively simple. Recall that step (ii) involves estimating the Fourier coefficient, $\hat{f}_\omega$, of each frequency $\omega \in (-N/2, N/2] \cap \mathbb{Z}$ identified during step (i). In the simplest case, this can be done for each such ω by using L independent and uniformly distributed random samples from f on $[0, 2\pi]$, $f(x_1), \ldots, f(x_L)$, in order to compute the estimator

$$A_\omega := \frac{1}{L} \sum_{l=1}^{L} f(x_l) \cdot e^{-\mathbb{i} \cdot \omega \cdot x_l}.$$

Note that A_ω is an unbiased estimator for $\hat{f}_\omega$ (i.e., $\mathbb{E}[A_\omega] = \hat{f}_\omega$) whose variance is $O(\|\hat{f}\|_2^2/L)$. Thus, A_ω will estimate $\hat{f}_\omega$ to high (relative) precision with high probability whenever $|\hat{f}_\omega|^2$ is relatively large compared to $\|\hat{f}\|_2^2/L$.

A naive implementation of step (*iii*) is even more straightforward. If

$$\{(\omega_m, A_{\omega_m}) \mid m \in [1, O(k)] \cap \mathbb{N}\}$$

is the sparse representation discovered during steps (*i*) and (*ii*) of the current repetition, then one simply replaces each as-yet-unused sample, $f(x)$, with

$$f(x) - \sum_{m=1}^{O(k)} A_{\omega_m} e^{\mathrm{i} \cdot \omega_m \cdot x}$$

during step (*iii*). These "updated samples" are then used in the subsequent repetitions of the three steps. See [19] for a more complete AAFFT tutorial, and [17,18,23] for the full details.

2.1.3 GFFT

The four GFFT variants are based on the algorithms introduced in [21,22, 3]. All four variants work by taking advantage of aliasing phenomena. More specifically, all four variants (implicitly) utilize a deterministic set of $K = O(k \log_k N)$ relatively prime numbers of size $O(k \log N) \ll N$.[1] Call these numbers $p_1, \ldots, p_K$. Each such p_j determines the length of an undersampled FFT of p_j equally spaced samples from f on $[0, 2\pi]$. Note the K resulting small FFTs are all highly aliased (since $p_1, \ldots, p_K$ are all smaller than N, the bandwidth of f). Thus, each FFT of length p_j effectively hashes the Fourier spectrum of f into a small array modulo p_j. The end result is that all K FFTs collectively bin the Fourier coefficients of f in a fashion similar to the one described above for AAFFT. However, in this case the strong combinatorial and number theoretic properties of $p_1, \ldots, p_K$ ensure that the frequency binning *always* separates the k most energetic Fourier coefficients from one another.

In fact, the combinatorial properties of the K aliased FFTs mentioned above are strong enough to allow accurate estimates to be computed for *every* Fourier coefficient of f. Both the slow variants of GFFT (i.e., GFFT-Det-Slow and GFFT-Rand-Slow) take advantage of these strong combinatorial properties in order to approximate $\hat{f}$ by: (*i*) estimating all N Fourier coefficients of f using (some of) the K aliased FFTs, (*ii*) sorting the Fourier coefficient estimates from step one by their magnitudes, and then (*iii*) outputting the $O(k)$ largest Fourier coefficient estimates together with their corresponding frequencies. The deterministic slow variant, GFFT-Det-Slow, uses all K aliased FFTs during step (*i*). The randomized slow variant, GFFT-Rand-Slow, only uses a small randomly selected subset of the K aliased FFTs during step (*i*). This randomly selected subset of the aliased FFTs maintains all the aforementioned

[1] See [3] for more information concerning the optimal choice of these relatively prime values.

strong combinatorial properties with high probability, however, thereby still ensuring accurate Fourier approximation with high probability.

Both of the fast variants of GFFT (i.e., GFFT-Det-Fast and GFFT-Rand-Fast) are obtained from the corresponding slow variants by employing a single round of energetic frequency identification before any estimation of Fourier coefficients is carried out. This ultimately saves time because Fourier coefficient estimation is then only performed for the relatively small number of energetic frequencies identified by this new first identification step, instead of for all frequencies as done by the slow variants. The identification step performed by the two fast GFFT variants is a number theoretic version of the identification step performed by AAFFT. Briefly, each fast variant performs a modified binary search within the frequency bins defined by the (subset of) the K aliased FFTs each uses for frequency coefficient estimation. These bins effectively isolate all energetic Fourier coefficients. Thus, the number theoretic binary search within each bin will tend to identify all sufficiently energetic frequencies. See section 1.1 of [21] for a more complete introduction to the basic ideas behind GFFT, and [21,22,3] for the full details.

3 Empirical Evaluation

In this section we carry out an empirical comparison of the implementations listed in Table 1. All experiments were run on a system with an AMD Phenom II X4 965 3.4 GHz processor and 8GB of DDR2-800 RAM, running Ubuntu 10.04 with the Linux Kernel 2.6.32-22 for x86_64, GCC 4.4.3 compiler, and the FFTW 3.2.2 library. FFTW3 was always run using an FFTW_PATIENT plan with wisdom enabled. A slightly improved version of AAFFT 0.9 from [23] was utilized as the AAFFT implementation. All GFFT variants were implemented in C++ as per Algorithms 2 and 3 in [22].[2] All run times are reported in tick counts using the "cycle.h" file included with the source code to the FFTW 3.2.2 library. Tick counts correspond closely to the number of CPU cycles used during the execution of a program and, therefore, accurately reflect each implementation's comparative computational complexity. They are used by FFTW3 in order to measure the run times of various FFT algorithms so as to select the fastest one for execution on problems of a given size (see [16] for details).

3.1 Runtime and Sampling Complexity

Every data point in Figures 1 through 4 has an associated (i) sparsity $k \in \{5, 20, 35, 50, 60, 65, 80, 95, 110\}$, ($ii$) bandwidth $N \in \{2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19},$

[2] Both the AAFFT and GFFT codes evaluated herein are freely available at `https://sourceforge.net/p/gopherfft/home/`.

2^{20}, 2^{21}, $2^{22}\}$, and (*iii*) set of 100 trial signals.[3] Every trial trigonometric polynomial, $f : [0, 2\pi] \to \mathbb{C}$, was constructed independently as follows: First, k frequency values, $\{\omega_1, \ldots, \omega_k\}$, were independently selected uniformly at random without replacement from $(-N/2, N/2] \cap \mathbb{Z}$. Next, k complex values, $\{C_1, \ldots, C_k\}$, were independently selected uniformly at random from the unit circle in the complex plane (i.e., each C_j has magnitude 1 and a random phase angle in $[0, 2\pi)$). Finally, f was defined to be

$$f(x) = \sum_{j=1}^{k} C_j \cdot e^{\mathbb{i} \cdot \omega_j \cdot x}. \tag{2}$$

Every data point plotted below in Figures 1 and 2 corresponds to the average runtime required by each Table 1 implementation to recover one of the data point's 100 associated trial trigonometric polynomials. Likewise, every data point plotted below in Figures 3 and 4 corresponds to the average number of function evaluations required by each Table 1 implementation to recover one of the data point's trial trigonometric polynomials. Furthermore, in the first four figures the upper and lower bars associated with each data point represent the maximum and minimum runtime/number of samples of each Table 1 implementation over all 100 associated trial signals, respectively. Thus, one can see from the figures below that all six implementations have runtime/sampling requirements for all 100 trial signals at each data point which generally concentrate tightly around their average value. This demonstrates that the randomized methods tested herein have predictable runtime/sampling requirements (i.e., that these values are not just "good on average", but rather "almost always good").

Note that when k is known in advance we can expect all four GFFT variants, when they succeed, to exactly recover each trial signal in Equation (2) (i.e., see Equation (1) in Theorem 2). On the other hand, AAFFT can only recover each signal up to a user defined precision (i.e., η in Theorem 1). Furthermore, some of the implementations in Table 1 guarantee recovery while others have some probability of failing to accurately recover each trial signal. In order to compare all six implementations despite these algorithmic differences, we chose parameters for each Monte-Carlo implementation (i.e., GFFT-rand-slow, GFFT-rand-fast, and AAFFT) which allowed it to recover every plotted data point's 100 trial signals with average ℓ_2-error below 0.05. However, the methods' actual average ℓ_2-errors were almost always substantially smaller.

In the experiments utilized to construct both Figures 1 and 3, AAFFT's average ℓ_2-error was below 10^{-7} for all data points (the maximum ℓ_2-error of any trial run was $< 2 \times 10^{-7}$). Similarly, all GFFT variants had average ℓ_2-errors below 10^{-8} for all data points in Figures 1 and 3, except for GFFT-Rand-Slow whose average ℓ_2-error was greater than 10^{-8} for the bandwidth

[3] We will let N refer to the bandwidth of f, as opposed to the degree of f, throughout Section 3 below. This slight abuse of notation will allow us to simplify the subsequent discussion while simultaneously preserving the correctness of the computational and sampling complexities listed in Table 1.

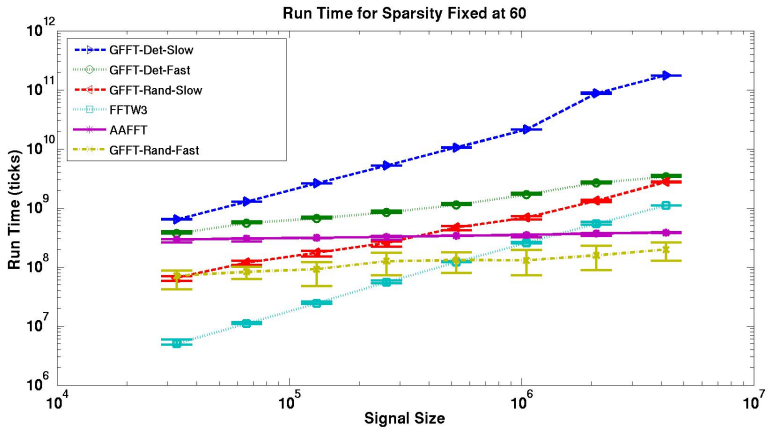

Fig. 1 Runtime of Implementations at Fixed Sparsity, $k = 60$

values of $N = 2^{20}$, 2^{21}, and 2^{22}. However, GFFT-Rand-Slow's average ℓ_2-error was still always below 0.04 for these bandwidth values.

In the experiments utilized to construct both Figures 2 and 4, AAFFT's average ℓ_2-error was below 3×10^{-6} for all data points (the maximum ℓ_2-error of any single trial run was $< 10^{-4}$). Similarly, all GFFT variants had average ℓ_2-errors below 10^{-8} for all data points in Figures 2 and 4, except for GFFT-Rand-Slow whose average ℓ_2-error was greater than 10^{-8} for the sparsity values of $k = 20, 35, 60$, and 110. GFFT-Rand-Slow's average ℓ_2-error was still always below 0.04 for these sparsity values, however. See Section 3.2 for additional discussion regarding the numerical accuracy of the various methods.

3.1.1 Runtime Complexity

Figure 1 compares the average runtime of the implementations in Table 1 when the sparsity, k, is fixed at 60.[4] The bandwidth of the trigonometric polynomial, or the signal size N, varies from 2^{15} to 2^{22} by powers of two. Using powers of two for N allows for the best possible performance by AAFFT (and FFTW3) against all four GFFT variants. FFTW3 is the fastest method for all signal sizes $N \leq 2^{19}$. However, for larger signal sizes GFFT-Rand-Fast (and later AAFFT) become faster than FFTW3. More importantly, the sublinear scaling of the fast methods with bandwidth, or signal size N, is clearly demonstrated. GFFT-Rand-Fast is further demonstrated to be the fastest of these for all bandwidth values.

Figure 2 compares the average runtime of all six implementations when the bandwidth, N, is fixed at 2^{22}.[5] The sparsity varies from $k = 5$ up to

[4] The runtime curves for other sparsity values look qualitatively similar.

[5] The bandwidth value of $N = 2^{22}$ was chosen to demonstrate the relative runtimes of AAFFT, GFFT, and FFTW3 for a bandwidth size on the order of one that might be encountered in the process of computing a three dimensional discrete Fourier transform with

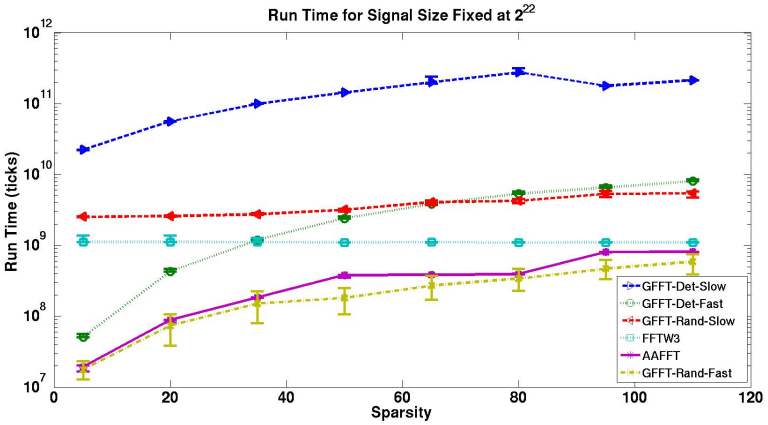

Fig. 2 Runtime of Implementations at Fixed Signal Size, $N = 2^{22}$

$k = 110$ by increments of 15. Looking at Figure 2 we can see that GFFT-Rand-Fast is again the fastest sublinear-time method (e.g., it generally outperforms AAFFT at all data points). At this bandwidth size GFFT-Rand-Fast is also faster than FFTW3 at recovering signals containing fewer than 110 nonzero Fourier coefficients. On the other hand, both slow GFFT variants take several times as long to finish recovery as FFTW3 does at each data point despite the fact that they require significantly fewer function samples (see Section 3.1.2 below). Runtime comparisons at other fixed sparsity and bandwidth values are qualitatively similar.

3.1.2 Sampling Complexity

Note that minimizing the runtime complexity of a Fourier approximation method necessitates a coinciding reduction in the method's sampling complexity. As a result, both AAFFT and all the GFFT variants discussed above have sublinear sampling complexities in addition to their sublinear runtime complexities.[6] In this section we empirically compare the sampling complexities of these algorithms. It is important to point out that the sampling requirements of the Fourier approximation methods considered herein are generally larger than existing compressed sensing methods (e.g., see the numerical experiments in [3]). However, sampling complexity might, as a secondary consideration, help one to decide to use one of the Fourier approximation algorithms mentioned above over another in some applications.

Figure 3 compares the average number of unique function evaluations, or function samples, taken by each implementation in Table 1 during the process

$\sim 10^2$ discritization points per dimension. The runtime curves for other bandwidth values look qualitatively similar.

[6] Explicit upper bounds on the sampling requirements of all GFFT variants are derived in [22].

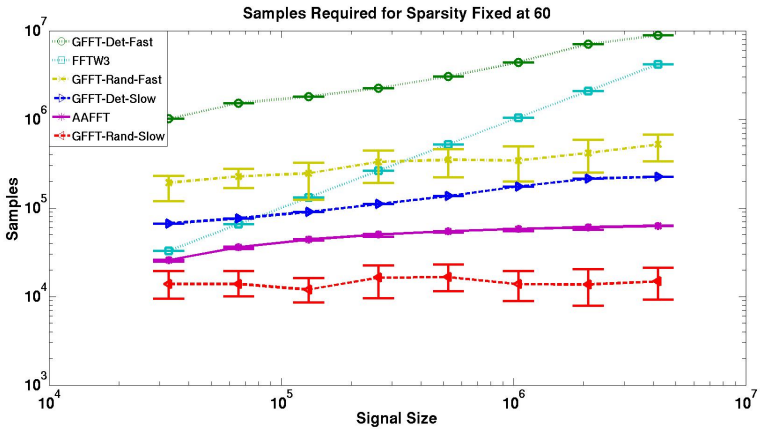

Fig. 3 Number of Function Evaluations Required at Fixed Sparsity, $k = 60$

of signal recovery when sparsity, k, is fixed at 60. The signal sizes vary as in Section 3.1.1 above. Looking at Figure 3 we can see that GFFT-Rand-Slow has the lowest sampling complexity for all signal sizes. However, AAFFT utilizes many fewer unique function evaluations than both fast variants of GFFT. It is important to point out, however, that AAFFT utilizes an *adaptive* sampling strategy which depends on the particular function being recovered. That is, the i^{th} point in $[0, 2\pi]$ at which AAFFT evaluates a given trial signal, $f : [0, 2\pi] \to \mathbb{C}$, depends on f's values at the preceding $i-1$ points. All GFFT variants, on the other hand, utilized *nonadaptive* samples which are completely independent of the trial function being approximated.

Figure 4 measures average sample usage with the signal bandwidths fixed to $N = 2^{22}$. The sparsity varies as in Section 3.1.1. The results again verify that GFFT-Rand-Slow generally has the lowest overall average sampling complexity. Likewise, AAFFT utilizes significantly fewer samples than both GFFT-Det-Fast and GFFT-Rand-Fast. Most importantly, we can see that all of GFFT-Det-Slow, AAFFT, and GFFT-Rand-Slow utilize significantly fewer samples than required by a traditional FFT in order to accurately recover the sparse signals considered in this section.

3.2 Numerical Accuracy and Approximation Error

In this section we investigate how the approximation accuracy of each of the Table 1 implementations deteriorates when the true/effective sparsity of the given signal is underestimated (i.e., when the sparsity parameter, k, is guessed/approximated incorrectly as the result of imperfect knowledge). Figure 5 represents such approximation error results for signals with bandwidth $N = 2^{22}$ as their true sparsity (i.e., number of nonzero Fourier coefficients) deviates from the sparsity parameter utilized by all of the Table 1 implementations. Every data point in Figure 5 has a different associated true sparsity

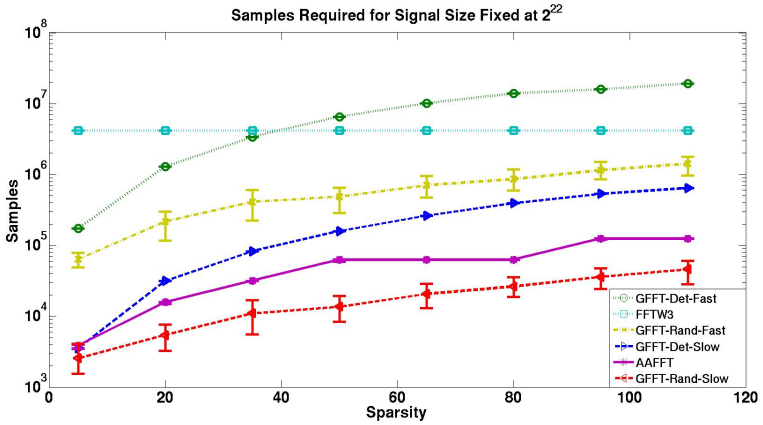

Fig. 4 Number of Function Evaluations Required at Fixed Signal Size, $N = 2^{22}$

value, $k_{\text{true}} \in \{20,\ 25,\ 30,\ 35,\ 40,\ 45,\ 50,\ 55,\ 60\}$, which was used to generate 1000 associated trial signals of that true Fourier sparsity. The Table 1 implementations, on the other hand, were run with their sparsity parameters set to 20 for all trial signals across all data points in Figure 5. Hence, the sparsity parameter utilized by each Fourier algorithm becomes increasingly inaccurate as the true signal sparsity increases (i.e., from left to right on the horizontal axis).

Every trial signal associated with a given data point, $f : [0, 2\pi] \to \mathbb{C}$, in Figure 5 was constructed independently as follows: First, k_{true} frequency values, $\{\omega_1, \ldots, \omega_{k_{\text{true}}}\}$, were independently selected uniformly at random without replacement from $(-N/2, N/2] \cap \mathbb{Z}$. Next, k_{true} complex values, $\{C_1, \ldots, C_{k_{\text{true}}}\}$, were independently selected uniformly at random from the unit circle in the complex plane (i.e., each C_j has magnitude 1 and a random phase angle in $[0, 2\pi)$). Finally, f was defined to be

$$f(x) = \sum_{j=1}^{k_{\text{true}}} C_j \cdot e^{i \cdot \omega_j \cdot x}. \tag{3}$$

Every data point plotted in Figure 5 corresponds to the average ℓ_2 approximation error made by each Table 1 implementation in recovering one of the data point's 1000 associated trial signals. The upper error bar associated with each data point represents the maximum ℓ_2 error made by each Table 1 implementation over all 1000 recovered trial signals. In generating Figure 5 the parameters of all of the Table 1 implementations were left the same as previously utilized to construct both Figures 2 and 4 for $k_{\text{true}} = k = k_{\text{est}} = 20$, with the exception of GFFT-Rand-Slow. The parameter used for GFFT-Rand-Slow was modified just enough to decrease its average ℓ_2 error of ≈ 0.04 from the

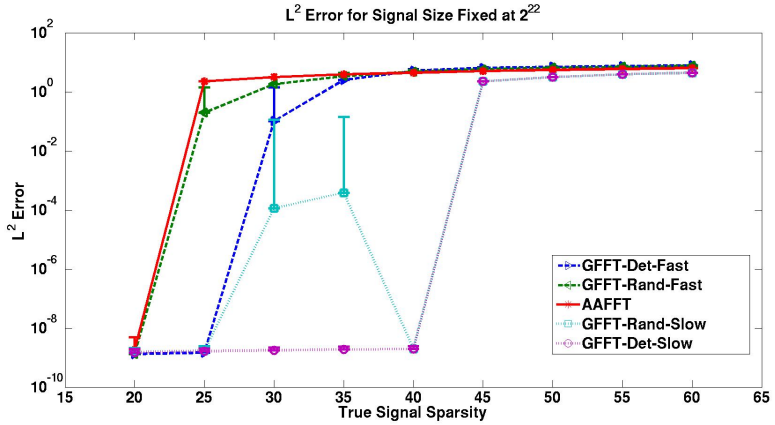

Fig. 5 The ℓ_2 error of the sparse Fourier representation returned by each Fourier algorithm (i.e., $\|\hat{f} - \mathbf{R}\|_2$). The signal size, N, was fixed at 2^{22}. Each Fourier algorithm utilized a sparsity parameter of $k_{\text{est}} = 20$. The true sparsity of each signal, k_{true}, varies between 20 and 60.

previous experiments at $k_{\text{true}} = k_{\text{est}} = 20$ to the same order of magnitude as the other methods.[7]

Looking at Figure 5 we see that GFFT-Det-Fast and GFFT-Det-Slow lose approximation accuracy more slowly than GFFT-Rand-Fast and GFFT-Rand-Slow, respectively, as the true signal sparsities deviate from 20. This is due primarily to the fact that they achieve their deterministic error guarantees (e.g., see Theorem 2) in these experiments by using enough functions samples/runtime to ensure that they will still likely produce accurate approximations of signals with true sparsity significantly larger than 20. Inspecting the randomized methods next, we see that GFFT-Rand-Slow generated the only nonmonotonically increasing error curve in Figure 5. This was due to it almost always producing highly accurate approximations as the true signal sparsity increased, except for a handful of signals with true sparsities of 30 and 35 for which it produced ℓ_2 errors of size ≈ 0.1.

As Figure 5 suggests, AFFT and GFFT-Rand-Fast are generally the least tolerant methods to incorrect sparsity estimates. AAFFT appears to be particularly sensitive – a defect of the method which could potentially be remedied by introducing additional procedures for adaptively tuning its many user specified parameters after each round of energetic frequency identification and Fourier coefficient estimation (see [23] for more details regarding AAFFT and its parameters). GFFT-Rand-Fast also quickly degrades in accuracy as the true signal sparsity deviates from its estimated sparsity value of 20, but has the relative benefit of having only two user specified parameters (including the

[7] GFFT-Rand-Slow only has one user specified runtime parameter besides the sparsity parameter, k. See [22] for details.

sparsity parameter, $k = k_{\text{est}}$).[8] This defect might also be remedied in the future by utilizing results along the lines of Theorem 4 below to determine if/when its two parameters should be modified in order to achieve a sufficiently small approximation error. However, we leave such modifications as future work.

4 Improved Theoretical Results

In this section an improved approximation result for Fourier compressible signals is discussed which retains the runtime complexity of Thoerem 2 (in expectation) while simultaneously reducing its sampling complexity. In addition to requiring only a near-optimal, $O(k \cdot \text{polylog}(N))$, number of samples (in the fashion of Theorem 1), the resulting method also exhibits a uniform error bound for all Fourier compressible signals (along the lines of Theorem 2). We begin by reviewing the theory necessary for the development of the improved algorithmic result presented later in Section 4.3.

4.1 Preliminaries: Fourier Compressible Sequences

Let $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_j, \ldots)$ be a complex sequence with finite ℓ_1-norm (i.e., with $\|\mathbf{a}\|_1 = \sum_{j=1}^{\infty} |\mathbf{a}_j| < \infty$). Given $\mathbf{a}$, there exists a sequence of natural numbers $\mathbf{j}_1, \mathbf{j}_2, \ldots, \mathbf{j}_l, \ldots$ which orders $\mathbf{a}$ so that $|\mathbf{a}_{\mathbf{j}_1}| \geq |\mathbf{a}_{\mathbf{j}_2}| \geq \ldots \geq |\mathbf{a}_{\mathbf{j}_l}| \geq \ldots$. We will refer to $\mathbf{a}$ as (b, ρ)-*compressible* if $|\mathbf{a}_{\mathbf{j}_l}| \leq b \cdot l^{-\rho}$ for all $l \in \mathbb{N}$.[9] We will say that a function, $f : [0, 2\pi] \to \mathbb{C}$, is Fourier (b, ρ)-compressible if its sequence of Fourier series coefficients is (b, ρ)-compressible.

It is not difficult to see that every sequence, $\mathbf{a}$, with finite ℓ_1-norm is $(\|\mathbf{a}\|_1, 1)$-compressible. Not surprisingly, for finite sequences we can say more. Suppose $\mathbf{a}$ contains only N nonzero entries. We can now ask what the largest ρ is which guarantees that $\mathbf{a}$ is $(2 \cdot \|\mathbf{a}\|_1, \rho)$-compressible (here the 2 can be replaced with any constant larger than one). To begin, suppose we choose ρ too large, so that there exists some $n \in [1, N] \cap \mathbb{Z}$ with $|\mathbf{a}_{\mathbf{j}_n}| > 2 \cdot \|\mathbf{a}\|_1 \cdot n^{-\rho}$. Then we have that

$$\|\mathbf{a}\|_1 \geq \sum_{l=1}^{n} |\mathbf{a}_{\mathbf{j}_l}| > 2 \cdot \|\mathbf{a}\|_1 \cdot n^{1-\rho}.$$

[8] It is important to note that GFFT-Rand-Fast requires additional parameters beyond the two referred to above to be chosen if one desires optimal performance. However, if the type of suboptimal performance demonstrated here is sufficient, these parameters can already be quickly estimated by automated procedures (e.g. by an optimized variant of Algorithm 3 in [21]). See [3] for a more detailed discussion regarding the optimal choice of these additional parameters.

[9] Note that a sequence $\mathbf{a}$ is (b, ρ)-compressible if and only if it is in the weak-$\ell_{1/\rho}$ ball of radius b. However, we will use the "ρ-compressible" notation which is more common in the computer science and algorithm orientated literature (e.g., see [12,32]). Those readers who are familiar with the relationship between weak and strong ℓ_p balls, etc., may skip to Section 4.2 below.

However, this can only be true if $\rho > 1 + \frac{1}{\log_2 n}$. The end result is that every finite sequence (i.e., vector) containing only N nonzero entries, $\mathbf{a} \in \mathbb{C}^N$, must be $\left(2 \cdot \|\mathbf{a}\|_1, 1 + \frac{1}{\log_2 N}\right)$-compressible. Hence, the set

$$\left\{ \mathbf{a} \;\middle|\; \mathbf{a} \text{ is } (2b, \rho)\text{-compressible for some } \rho > 1 \right\}$$

contains all finite sequences in the ℓ_1-ball of radius b. Of course, for the purposes of sparse approximation we are primarily interested in (b, ρ)-compressible signals which have high fidelity sparse representations under the ℓ_1-norm (e.g., signals which are compressible with $\rho \geq 2$).

Many functions $f : [0, 2\pi] \to \mathbb{C}$ exhibit Fourier compressibility. For example, any 2π-periodic function with an integrable second derivative is Fourier (c, ρ)-compressible with $\rho \geq 2$ for some constant $c \in \mathbb{R}^+$ (see [5]). See [29] and the references therein for more about the sparse approximation of (Fourier) compressible signals.

Given a Fourier (b, ρ)-compressible function, $f : [0, 2\pi] \to \mathbb{C}$, we would like to approximate its sequence of Fourier series coefficients using k of its largest magnitude coefficients. Approximating $\hat{f}$ by $\mathbf{R}_{\text{opt}(k)}$ leads to approximation errors of size

$$\|\hat{f} - \mathbf{R}_{\text{opt}(k)}\|_1 = \sum_{l=k+1}^{N} \left|\hat{f}_{\mathbf{j}_l}\right| \leq \int_k^\infty b\ y^{-\rho}\ dy \ = \ \frac{b}{\rho - 1} \cdot k^{1-\rho}$$

and

$$\|\hat{f} - \mathbf{R}_{\text{opt}(k)}\|_2 \leq \sqrt{\int_k^\infty b^2\ y^{-2\rho}\ dy} = \frac{b}{\sqrt{2\rho - 1}} \cdot k^{\frac{1}{2}-\rho}.$$

If ρ is greater than 1 we can see these errors are bounded. Indeed, they quickly become manageable as ρ, k increase. Thus, we may provide uniform sparse approximation bounds for classes of Fourier compressible signals whenever we have knowledge regarding ρ and b (i.e., the functions' smoothness).

4.2 Preliminaries: The Restricted Isometry Property

Many compressed sensing (CS) methods (e.g., [6–8,37,25,32–34,4]) deal with recovering the k most significant entries of a vector, $\mathbf{a} \in \mathbb{C}^N$, using only a compressed set of linear measurements given by $\mathcal{M}\mathbf{a}$, where $\mathcal{M}$ is a special type of rectangular $K \times N$ matrix with the *Restricted Isometry Property (RIP)*. A matrix $\mathcal{M}$ has the RIP(N,k,ϵ) if

$$(1 - \epsilon)\|\mathbf{a}\|_2^2 \leq \|\mathcal{M}\mathbf{a}\|_2^2 \leq (1 + \epsilon)\|\mathbf{a}\|_2^2 \tag{4}$$

for all $\mathbf{a} \in \mathbb{C}^N$ containing at most k non-zero coordinates. Let K be $\Omega(k \cdot \log(N/k))$. A $K \times N$ RIP(N,k,$\Omega(1)$) matrix may be constructed with high probability by independently choosing each $\mathcal{M}_{i,j}$ entry via a 0-mean, $\frac{1}{K}$-variance Gaussian distribution [8]. Furthermore, these Gaussian matrices are

near-optimal (i.e., they have a near-minimal number of rows while still allowing the best k-term approximation [10] of $\mathbf{a}$ given $\mathcal{M}\mathbf{a}$). Any $K \times N$ RIP(N,k,$\Omega(1)$) matrix must have $K = \Omega(k \cdot \log(N/k))$. However, given that we are primarily interested in Fourier recovery, we will focus on the following RIP matrix construction theorem proven in [36].

Theorem 3 ***(Rudelson, Vershynin).*** *Suppose we select K rows uniformly at random from the rescaled $N \times N$ Inverse Discrete Fourier Transform (IDFT) matrix $\frac{2\pi}{\sqrt{K \cdot N}} \Psi^{-1}$, where*

$$\left(\Psi^{-1}\right)_{i,j} = \frac{e^{\frac{2\pi \mathbf{i} \cdot i \cdot j}{N}}}{2\pi},$$

and form a $K \times N$ matrix, $\mathcal{M}$. If K is $\Omega\left(k \cdot \log N \cdot \log^2 k \cdot \log(k \log N)\right)$ then $\mathcal{M}$ will have the RIP(N,k,$\Omega(1)$) with high probability.

Let $\mathcal{M}$ be a RIP(N,$O(k)$,ϵ) matrix with $\epsilon \in \left(0, \frac{1}{2}\right)$ sufficiently small. In this case many different methods (e.g., convex optimization techniques [6,8]) can recover a near optimal $O(k)$-term representation for an unknown $\mathbf{a} \in \mathbb{C}^N$ given only the compressed vector $\mathcal{M}\mathbf{a} \in \mathbb{C}^K$ as input. Therefore, if $\mathbf{a}$ is taken to be the Fourier transform of a band-limited and Fourier compressible signal, f, Theorem 3 implies that only a small set of $O\left(k \cdot \log^4 N\right)$ random samples from f are necessary in order to accurately approximate $\mathbf{a} = \hat{f}$.

Besides facilitating sparse recovery, RIP matrices can also provide accurate energy estimates for compressible signals. If f is Fourier (b, ρ)-compressible with bandwidth N, we can represent it as

$$\hat{f} = \mathbf{R}_{\text{opt}(k)} + \mathbf{r}, \tag{5}$$

where $\mathbf{R}_{\text{opt}(k)}$ is an optimal k-sparse representation for $\hat{f}$, and $\|\mathbf{r}\|_2 \leq \frac{b}{\sqrt{2\rho - 1}} \cdot k^{\frac{1}{2}-\rho}$. Thus, $\hat{f} \approx \mathbf{R}_{\text{opt}(k)}$ for k sufficiently large. In this case Equation 4 suggests that a RIP matrix $\mathcal{M}$ might have $\|\mathcal{M}\hat{f}\|_2 \approx \|\hat{f}\|_2$ which would allow us to estimate the energy in $\hat{f}$ using only the compact sketch $\mathcal{M}\hat{f} \in \mathbb{C}^K$ given by randomly sampling f inside $[0, 2\pi]$ as per Theorem 3. More importantly, we shall see that this observation allows us to quickly estimate how well any given sparse representation approximates a compressible $\hat{f} \in \mathbb{C}^N$. We have the following theorem.

Theorem 4 *Let $\tau \in (1, \infty)$, $\mathcal{M}$ be a $K \times N$ matrix with RIP(N,$2k$,ϵ), and $\hat{f} \in \mathbb{C}^N$ be (b, ρ)-compressible. If $\boldsymbol{R}$ is a k-term sparse representation for $\hat{f}$ with $\|\hat{f} - \boldsymbol{R}\|_2 \leq \tau$ then*

$$\|\mathcal{M}(\hat{f} - \boldsymbol{R})\|_2 \leq \sqrt{1+\epsilon} \cdot \tau + 2b \cdot k^{\frac{1}{2}-\rho} \sqrt{1+\epsilon} \left(\frac{1}{\sqrt{2\rho - 1}} + \frac{1}{2\sqrt{2}(\rho - 1)} \right). \tag{6}$$

Whenever $\|\mathcal{M}(\hat{f} - \boldsymbol{R})\|_2 \leq \tilde{\tau}$ then

$$\|\hat{f} - \boldsymbol{R}\|_2 \leq \frac{\tilde{\tau}}{\sqrt{1-\epsilon}} + b \cdot k^{\frac{1}{2}-\rho} \left[\frac{1}{\sqrt{2\rho - 1}} \left(1 + \frac{\sqrt{1+\epsilon}}{\sqrt{1-\epsilon}} \right) + \frac{\sqrt{1+\epsilon}}{\sqrt{1-\epsilon}} \frac{1}{\sqrt{2}(\rho - 1)} \right].$$

Proof:

We give an argument similar to ones used in [32] to establish Theorems A.1 and A.2. Let $e = \frac{b}{\sqrt{2\rho-1}} k^{\frac{1}{2}-\rho}$ and define $\mathbf{r} = \hat{f} - \mathbf{R}_{\text{opt}(k)}$ as per Equation 5. Assuming that $\|\hat{f} - \mathbf{R}\|_2 \leq \tau$ we have

$$\|\mathbf{R}_{\text{opt}(k)} - \mathbf{R}\|_2 \leq \|\hat{f} - \mathbf{R}\|_2 + \|\mathbf{r}\|_2 \leq \tau + e.$$

Therefore,

$$\begin{aligned}
\|\mathcal{M}(\hat{f} - \mathbf{R})\|_2 &\leq \|\mathcal{M}(\mathbf{R}_{\text{opt}(k)} - \mathbf{R})\|_2 + \|\mathcal{M}\mathbf{r}\|_2 \\
&\leq \sqrt{1+\epsilon} \cdot (\tau + e) + \|\mathcal{M}\mathbf{r}\|_2 \quad \text{(see Equation 4)} \\
&\leq \sqrt{1+\epsilon} \cdot \left(\tau + 2e + \frac{\|\mathbf{r}\|_1}{\sqrt{2k}}\right) \quad \text{(see Proposition 3.5 in [32])} \\
&\leq \sqrt{1+\epsilon} \cdot \left(\tau + 2e + \frac{b \cdot k^{\frac{1}{2}-\rho}}{\sqrt{2}(\rho-1)}\right).
\end{aligned}$$

Substituting for e and collecting terms we arrive at Equation 6.

Assuming $\|\mathcal{M}(\hat{f} - \mathbf{R})\|_2 \leq \tilde{\tau}$, we have

$$\begin{aligned}
\|\mathcal{M}(\hat{f} - \mathbf{R})\|_2 &\geq \|\mathcal{M}(\mathbf{R}_{\text{opt}(k)} - \mathbf{R})\|_2 - \|\mathcal{M}\mathbf{r}\|_2 \\
&\geq \sqrt{1-\epsilon} \cdot \|\mathbf{R}_{\text{opt}(k)} - \mathbf{R}\|_2 - \|\mathcal{M}\mathbf{r}\|_2 \quad \text{(see Equation 4)}
\end{aligned} \tag{7}$$

which implies that

$$\|\mathbf{R}_{\text{opt}(k)} - \mathbf{R}\|_2 \leq \frac{\tilde{\tau} + \|\mathcal{M}\mathbf{r}\|_2}{\sqrt{1-\epsilon}}.$$

Therefore, we have

$$\begin{aligned}
\|\hat{f} - \mathbf{R}\|_2 \leq \|\mathbf{R}_{\text{opt}(k)} - \mathbf{R}\|_2 + \|\mathbf{r}\|_2 &\leq \frac{\tilde{\tau} + \|\mathcal{M}\mathbf{r}\|_2}{\sqrt{1-\epsilon}} + \|\mathbf{r}\|_2 \\
&\leq \frac{\tilde{\tau} + \sqrt{1+\epsilon} \cdot \left(\|\mathbf{r}\|_2 + \frac{\|\mathbf{r}\|_1}{\sqrt{2k}}\right)}{\sqrt{1-\epsilon}} + \|\mathbf{r}\|_2 \quad \text{(see Proposition 3.5 in [32])}.
\end{aligned}$$

Bounding $\|\mathbf{r}\|_2$ and $\|\mathbf{r}\|_1$ and then collecting terms we obtain our result. □

Theorem 4 tells us that we can use a RIP matrix $\mathcal{M}$ to verify the quality of any sparse representation for $\hat{f}$ that we are lucky enough to find. Furthermore, if vector products with $\mathcal{M}$ can be computed quickly, we will have a fast method for checking sparse representations we generate. We will ultimately use this idea to convert fast Monte Carlo Fourier results which occasionally fail to accurately approximate Fourier compressible functions into Las Vegas Fourier results which are always guaranteed to accurately approximate Fourier compressible functions, but are sometimes slow. With this idea in mind we begin to consider new Fourier results.

Algorithm 1 LAS VEGAS FOURIER

Input: Theorem 3 RIP(N,$2k$,$\Omega(1)$) matrix $\mathcal{M}$, Fourier (b,ρ)-compressible f, $\eta, \epsilon \in (0,1/2)$
Output: $2k$-sparse representation $\mathbf{R}$
$\mathbf{R} \leftarrow$ Call Corollary 4 in Table 1 with failure probability $< 1/N$, and tolerance ϵ
$\mathbf{y} \leftarrow \mathcal{M}\hat{f}$ (i.e., randomly sample f in $[0,2\pi]$)
if $\|\mathbf{y} - \mathcal{M}\mathbf{R}\|_2 > b \cdot k^{\frac{1}{2}-\rho}\left(\frac{3\sqrt{2}}{\sqrt{2\rho-1}} + \frac{1+\sqrt{2}}{\rho-1}\right)$ **then**
    $\mathbf{R} \leftarrow$ CoSaMP [32] with input $\mathcal{M}$, η, $\mathbf{y}$
end if
Output $\mathbf{R}$

4.3 Improved Approximation Guarantees

In this section we present a fast randomized Fourier method which is guaranteed (in the presence of Theorem 3 RIP matrices) to return high quality Fourier representations using nonadaptive sublinear-sampling. Most importantly, the presented method also runs in sublinear-time with high probability on each input signal. In order to achieve this result we use a simple "guess and check" approach. We first run a fast randomized method (e.g., see the Theorem 1 and Corollary 4 rows of Table 1) to quickly recover a sparse Fourier representation which is likely to be of high quality. We then check to see if the returned representation is accurate or not using a RIP matrix (see Theorem 4). If the representation is good, we return it as our answer. Otherwise, in the unlikely event that the randomized approach has failed, we call a superlinear-time method (e.g., any of those discussed in [6–8,37,25,32–35,4]) to calculate a good answer.[10] See Algorithm 1 for pseudocode.

It is important to note that the RIP matrices used by Algorithm 1 do not need to be instantiated. For the purposes of Algorithm 1 it is sufficient to only know the randomly selected IDFT matrix row numbers used to construct a Theorem 3 RIP matrix. Furthermore, as written, Algorithm 1 only requires nonadaptive sampling access to f. Keeping these points in mind we obtain the following Theorem.

Theorem 5 *Fix a precision parameter $\eta \in \mathbb{R}^+$ and let $\mathcal{M}$ be a Fourier RIP(N,$2k$,$\Omega(1)$) matrix as per Theorem 3. Then, given sampling access to any Fourier (b,ρ)-compressible function $f : [0,2\pi] \to \mathbb{C}$ with bandwidth N, Algorithm 1 will produce a $2k$-sparse representation $\mathbf{R}$ with ℓ_2-error, $\|\hat{f} - \mathbf{R}\|_2$, that is*

$$O\left(b \cdot \left(1 + \frac{1}{\rho - 1}\right) \cdot k^{\frac{1}{2}-\rho} + \eta\right).$$

The number of required samples from f is guaranteed to be $O\left(k \cdot \log^4 N\right)$. The runtime of Algorithm 1 is always guaranteed to be

$$O\left(N \cdot \log N \cdot \log(\|\hat{f}\|_2/\eta) + (k + \log N) \cdot k \cdot \log^4 N\right).$$

[10] We will utilize CoSaMP [32] as the superlinear-time method below due to its fast runtime complexity in the Fourier setting.

The expected runtime is $O\left((k+\log N)\cdot k\cdot\log^4 N+\log N\cdot\log(\|\hat{f}\|_2/\eta)\right)$.

Proof:

We begin by verifying the error guarantee: In the unlikely event that the superlinear-time method is called, the error guarantee is inherited from Theorem A in [32]. If the superlinear-time method is not called, then

$$\|\mathcal{M}\hat{f}-\mathcal{M}\mathbf{R}\|_2\leq b\cdot k^{\frac{1}{2}-\rho}\left(\frac{3\sqrt{2}}{\sqrt{2\rho-1}}+\frac{1+\sqrt{2}}{\rho-1}\right). \tag{8}$$

However, in this case Theorem 4 once again guarantees the stated error bound. The guaranteed sample bound comes from the fact that both Corollary 4 in [22] and Theorem 3 are utilized only once.

We now consider the runtime requirements. Note that $\mathcal{M}\mathbf{R}$ can be calculated exactly in $O\left(k^2\cdot\log^4 N\right)$ time (see line 5). This fact combined with the runtime bounds from Theorem A in [32] (see line 6) and Corollary 4 in [22] (see line 3) yields the stated worst case runtime complexity. We conclude by analyzing the expected runtime.

The Monte Carlo method associated with Corollary 4 in Table 1 that is utilized in line 3 of Algorithm 1 will produce a sparse Fourier representation, $\mathbf{R}$, that fails to satisfy Equation 1 with probability less than $1/N$. Furthermore, whenever the sparse representation produced by line 3 does satisfy Equation 1, we have

$$\|\hat{f}-\mathbf{R}\|_2\leq b\cdot k^{\frac{1}{2}-\rho}\left(\frac{1}{\sqrt{2\rho-1}}+\frac{1}{\rho-1}\right)$$

which will in turn guarantee that Equation 8 holds by Theorem 4. Hence, the superlinear-time method in line 6 of Algorithm 1 will be run with probability less than $1/N$. The expected runtime now follows easily from Theorem A in [32], Corollary 4 in [22], and the fact that $\mathcal{M}\mathbf{R}$ can be calculated exactly in $O\left(k^2\cdot\log^4 N\right)$ time. □

Note that Theorem 5 can be viewed as an existence result. In essence, it indicates that Las Vegas results with sublinear runtime and uniform approximation guarantees along the lines of Theorem 2 can be achieved using only $O(k\cdot\log^4 N)$ nonadaptive samples per signal, instead of $O(k^2\cdot\log^4 N)$ samples. Having the matrix, $\mathcal{M}$, from line 1 of Algorithm 1 be a Fourier RIP(N,$2k$,$\Omega(1)$) matrix is all that is required. Furthermore, Theorem 3 promises that one can obtain such a RIP matrix with high probability (i.e., Theorem 3 constructively demonstrates that such matrices exist and are, in fact, very common).

However, although Theorem 5 does achieve error and runtime bounds similar to those of Theorem 2 while requiring fewer samples, it is important to point out that both the runtime and approximation guarantees of Theorem 5 are slightly weaker than those of Theorem 2. First, the sublinear runtime guarantee for Theorem 5 holds only with high probability (and, therefore, in expectation). Theorem 2, on the other hand, guarantees that an essentially

identical runtime bound will hold both uniformly and deterministically for all signals. Secondly, and more importantly, the uniform approximation guarantee provided by Theorem 5 is a min-max error guarantee over the class of Fourier (b, ρ)-compressible functions. This type of error guarantee is strictly weaker than the type of instance optimal approximation results provided by both Theorems 1 and 2 which guarantee that the output sparse representations will be nearly optimal with respect to every individual input function f. Among other things, knowledge regarding the b and ρ parameters of each given input function f (i.e., knowledge regarding f's smoothness) is required for the approximation result of Theorem 5 to be useful in practice. Theorem 2, on the other hand, guarantees a near optimal sparse approximation for every input function, f, in any case.

5 Conclusion

In this paper we implemented and empirically evaluated the sparse Fourier transform algorithms proposed in [22]. Our implementation, the Gopher Fast Fourier Transform (GFFT), was bench marked against both AAFFT [23], an earlier sparse Fourier transform algorithm, and FFTW3 [16], a highly optimized standard FFT implementation. As indicated in Section 3, the GFFT-Rand-Slow variant of GFFT generally has the lowest sampling complexity of all the tested implementations. Similarly, the GFFT-Rand-Fast variant of GFFT generally has the lowest runtime complexity of all the tested implementations.

In addition to carrying out an empirical evaluation of GFFT, we also presented a fast sparse Fourier transform method having both uniform error guarantees over the class of compressible signals, and near optimal sampling complexity. This represents an improvement in sampling requirements over previous fast methods with uniform error guarantees (e.g., see Theorem 2). However, despite the fact this method obtains $O(k \cdot \log^4 N)$ sampling complexity, it is worth noting that it remains roughly $O(k^2 \cdot \log^5 N)$-time (i.e., its runtime complexity still scales quadratically in the sparsity parameter k). Hence, it does not represent a strict improvement over previous faster Monte Carlo results with nonuniform error guarantees (e.g., see Theorem 1). This defect can be overcome if the sparse matrix-vector product required in line 5 of Algorithm 1, $\mathcal{M}\mathbf{R}$, can be computed faster than possible by naive $O\left(k^2 \cdot \log^4 N\right)$-time sparse matrix multiplication. However, this appears to be difficult in general as neither the frequencies in $\mathbf{R}$ nor the IDFT rows composing $\mathcal{M}$ (see Theorem 3) need have any of the structure necessary for the application of standard unequally spaced FFT methods [15].

Acknowledgments

We would like to thank Martin Strauss for answering questions concerning Theorem 1. We would also like to thank Martin Strauss, Tsvetanka Sendova, and Mark Herman for helpful discussions.

References

1. Akavia, A.: Deterministic sparse fourier approximation via fooling arithmetic progressions. 23rd Conference on Learning Theory (2010)
2. Akavia, A., Goldwasser, S., Safra, S.: Proving hard-core predicates using list decoding. Annual Symposium on Foundations of Computer Science **44**, 146–159 (2003)
3. Bailey, J., Iwen, M.A., Spencer, C.V.: On the design of deterministic matrices for fast recovery of fourier compressible functions. SIAM Journal on Matrix Analysis and Applications **33**(1), 263–289 (2012)
4. Blumensath, T., Davies, M.E.: Iterative hard thresholding for compressed sensing. Applied and Computational Harmonic Analysis **27**(3), 265 – 274 (2009)
5. Boyd, J.P.: Chebyshev and Fourier Spectral Methods. Dover Publications, Inc. (2001)
6. Candes, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inform. Theory **52**, 489–509 (2006)
7. Candes, E., Romberg, J., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Communications on Pure and Applied Mathematics **59**(8), 1207–1223 (2006)
8. Candes, E., Tao, T.: Near optimal signal recovery from random projections: Universal encoding strategies? IEEE Trans. on Information Theory (2006)
9. Chazelle, B.: The Discrepancy Method: Randomness and Complexity. Brooks/Cole Publishing Company (1992)
10. Cohen, A., Dahmen, W., DeVore, R.: Compressed Sensing and Best k-term Approximation. Journal of the American Mathematical Society **22**(1), 211–231 (2008)
11. Cooley, J., Tukey, J.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. **19**, 297–301 (1965)
12. Cormode, G., Muthukrishnan, S.: Combinatorial algorithms for compressed sensing. Structural Information and Communication Complexity pp. 280–294 (2006)
13. Daubechies, I., Runborg, O., Zou, J.: A sparse spectral method for homogenization multiscale problems. Multiscale Model. Sim. (2007)
14. Donoho, D.: Compressed Sensing. IEEE Trans. on Information Theory **52**, 1289–1306 (2006)
15. Dutt, A., Rokhlin, V.: Fast Fourier transforms for nonequispaced data. SIAM J. Sci. Comput. **14**, 1368–1383 (1993)
16. Frigo, M., Johnson, S.: The design and implementation of fftw3. Proceedings of IEEE 93 (2) pp. 216–231 (2005)
17. Gilbert, A., Guha, S., Indyk, P., Muthukrishnan, S., Strauss, M.: Near-optimal sparse Fourier estimation via sampling. ACM STOC pp. 152–161 (2002)
18. Gilbert, A., Muthukrishnan, S., Strauss, M.: Improved time bounds for near-optimal sparse Fourier representations. Proceedings of SPIE Wavelets XI (2005)
19. Gilbert, A., Strauss, M., Tropp, J.: A tutorial on fast fourier sampling. Signal Processing Magazine, IEEE **25**(2), 57–66 (2008)
20. Gilbert, A.C., Strauss, M.J., Tropp, J.A., Vershynin, R.: One sketch for all: Fast algorithms for compressed sensing. Proc. 39^{th} ACM Symp. Theory of Computing (2007)
21. Iwen, M.A.: Combinatorial sublinear-time fourier algorithms. Foundations of Computational Mathematics **10**(3), 303 – 338 (2010)
22. Iwen, M.A.: Improved approximation guarantees for sublinear-time fourier algorithms. Applied and Computational Harmonic Analysis (to appear)
23. Iwen, M.A., Gilbert, A.C., Strauss, M.J.: Empirical evaluation of a sub-linear time sparse DFT algorithm. Communications in Mathematical Sciences **5**(4) (2007)
24. Kirolos, S., Laska, J., Wakin, M., Duarte, M., Baron, D., Ragheb, T., Massoud, Y., Baraniuk, R.: Analog-to-information conversion via random demodulation. Proc. IEEE Dallas Circuits and Systems Conference (2006)
25. Kunis, S., Rauhut, H.: Random Sampling of Sparse Trigonometric Polynomials II - Orthogonal Matching Pursuit versus Basis Pursuit. Foundations of Computational Mathematics **8**(6), 737–763 (2008)
26. Kushilevitz, E., Mansour, Y.: Learning decision trees using the fourier spectrum. SICOMP **22**(6), 1331–1348 (1993)

27. Laska, J., Kirolos, S., Massoud, Y., Baraniuk, R., Gilbert, A., Iwen, M., Strauss, M.: Random sampling for analog-to-information conversion of wideband signals. Proc. IEEE Dallas Circuits and Systems Conference (2006)
28. Lustig, M., Donoho, D., Pauly, J.: Sparse MRI: The application of compressed sensing for rapid MR imaging. Magnetic Resonance in Medicine **58**(6), 1182–1195 (2007)
29. Mallat, S.: A Wavelet Tour of Signal Processing. Academic Press (2003)
30. Mansour, Y.: Learning boolean functions via the fourier transform. Theoretical Advances in Neural Computation and Learning pp. 391–424 (1994)
31. Mansour, Y.: Randomized approxmation and interpolation of sparse polynomials. SIAM Journal on Computing **24:2** (1995)
32. Needell, D., Tropp, J.A.: Cosamp: Iterative signal recovery from incomplete and inaccurate samples. Applied and Computational Harmonic Analysis **26**(3) (2008)
33. Needell, D., Vershynin, R.: Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. Foundations of Computational Mathematics **9**, 317–334 (2009)
34. Needell, D., Vershynin, R.: Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. IEEE Journal of Selected Topics in Signal Processing **4**(2), 310–316 (2010)
35. Rauhut, H.: Compressive sensing and structured random matrices. Theoretical Foundations and Numerical Methods for Sparse Recovery pp. 1–92 (vol. 9 in Radon Series Comp. Appl. Math., deGruyter, 2010)
36. Rudelson, M., Vershynin, R.: Sparse reconstruction by convex relaxation: Fourier and gaussian measurements. In: 40th Annual Conference on Information Sciences and Systems (CISS) (2006)
37. Tropp, J., Gilbert, A.: Signal recovery from partial information via orthogonal matching pursuit. IEEE Trans. Info. Theory **53**(12), 4655–4666 (2007)