A Deterministic Sparse FFT for Functions with Structured Fourier Sparsity

Sina Bittens[*], Ruochuan Zhang[†], Mark A. Iwen[‡]

May 16, 2017

Abstract

In this paper a deterministic sparse Fourier transform algorithm is presented which breaks the quadratic-in-sparsity runtime bottleneck for a large class of periodic functions exhibiting structured frequency support. These functions include, e.g., the oft-considered set of block frequency sparse functions of the form

$$f(x) = \sum_{j=1}^{n} \sum_{k=0}^{B-1} c_{\omega_j+k} e^{i(\omega_j+k)x}, \quad \{\omega_1, \ldots, \omega_n\} \subset \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor \right] \cap \mathbb{Z}$$

as a simple subclass. Theoretical error bounds in combination with numerical experiments demonstrate that the newly proposed algorithms are both fast and robust to noise. In particular, they outperform standard sparse Fourier transforms in the rapid recovery of block frequency sparse functions of the type above.

AMS Subject Classification. 05-04, 42A10, 42A15, 42A16, 42A32, 65T40, 65T50, 68W25, 94A12

1 Introduction

In this paper we consider the problem of deterministically recovering a periodic function $f\colon [0, 2\pi] \to \mathbb{C}$ as rapidly as absolutely possible via sampling. In particular, we focus on a specific set of functions f whose dominant Fourier series coefficients are all associated with frequencies contained in a small number, n, of unknown structured support sets $S_1, \ldots, S_n \subset (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$, where $N \in \mathbb{N}$ is very large. In such cases the function f will have the form

$$f(x) \approx \sum_{j=1}^{n} \sum_{\omega \in S_j} c_\omega e^{i\omega x}, \tag{1}$$

where each unknown S_j has simplifying structure (e.g., has $|x - y| < B \ll N$ for all $x, y \in S_j$).

[*]University of Göttingen, Institute for Numerical and Applied Mathematics, Lotzestr. 16-18, 37083 Göttingen, Germany (sina.bittens@mathematik.uni-goettingen.de).

[†]Department of Mathematics, Michigan State University, East Lansing, MI, 48824, USA (zhangr12@msu.edu).

[‡]Department of Mathematics, and Department of Computational Mathematics, Science, and Engineering (CMSE), Michigan State University, East Lansing, MI, 48824, USA (markiwen@math.msu.edu).

The classical solution for this problem would be to compute the Discrete Fourier Transform (DFT) of N equally spaced samples from f on $[0, 2\pi]$, $\left(f\left(\frac{2\pi j}{N}\right)\right)_{j=0}^{N-1}$, in order to obtain approximations of c_ω for all $\omega \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ in $\mathcal{O}(N \log N)$-time. Herein, we instead consider faster deterministic Sparse Fourier Transform (SFT) methods which are guaranteed to recover such f using a number of samples and operations that scale at most polynomially in both $\sum_{j=1}^{n} |S_j|$ and $\log(N)$. Such algorithms will always be faster than classical $\mathcal{O}(N \log(N))$-time methods whenever the cardinalities of the support sets, $|S_j|$, are sufficiently small in comparison to N. The main contribution of this paper is the development of the fastest known deterministic SFT methods to date for the recovery of a large class of periodic functions of type (1). Such functions (1) will be referred to as functions with *structured frequency support* below.

1.1 Related Work: Sparse Fourier Transforms

The vast majority of the work on sparse Fourier transform methods has focused on the unstructured frequency sparse case where, e.g., each set S_j in (1) is just a singleton set. In this case functions of the form (1) are simply n-sparse in the Fourier domain. The first sub-linear time methods developed for rapidly computing the Fourier series coefficients of such frequency sparse functions were randomized algorithms [2, 13, 15, 30] which fail to output good solutions with some constant (and usually tunable) probability. In exchange for this slight unreliability in producing accurate output, the fastest of these randomized techniques are able to compute the Fourier series of n-sparse f in just $n \log^{\mathcal{O}(1)} N$-time. The most efficient, numerically stable, and publicly available implementations of these methods are based on random algorithms developed out of MIT [17, 19, 22], Michigan [13, 15, 26], and Michigan State [10, 31, 41].[1] We point the reader to a recent survey of such algorithms, techniques, and implementations for more details [14].

Herein we are interested in deterministic SFT methods with no probability of failing to recover the dominant Fourier series coefficients of f. As with randomized techniques, most methods of this kind (see, e.g., [1, 23, 24, 37]) focus on the recovery of functions f that are unstructured and (approximately) n-sparse in the Fourier domain. As one might expect, these techniques are generally slower than their randomized counterparts, and the fastest run in $n^2 \log^{\mathcal{O}(1)} N$ time in the unstructured frequency sparse case.

Note the quadratic runtime of these deterministic methods in n. It is worth mentioning that reducing the quadratic runtime dependence on n for unstructured frequency sparse signals necessitates a similar reduction in the sampling complexity of these deterministic methods which (even when considered independently of the sub-linear runtimes we demand herein) is known to be notoriously difficult (see, e.g., [7, 9, 12]). This makes meaningful runtime reductions of these methods for periodic functions with unstructured sparsity quite unlikely to occur anytime soon. However, runtime reductions for functions with structured frequency sparsity should be more tractable. In this paper we demonstrate this fact by constructing deterministic algorithms which achieve sub-linear runtimes that scale sub-quadratically in sparsity for a wide class of functions with structured frequency support.

[1] The code for all of these implementations is freely available on the web [18, 25].

1.2 A General Class of Functions with Structured Frequency Support

Existing sparse Fourier transform techniques have been applied to many signal processing problems including, e.g., GPS signal acquisition [16], analog-to-digital conversion [29,43], and wideband communication/spectrum sensing [20, 42]. In all of these applications the signals under consideration are generally manmade and, therefore, structured in Fourier space. Herein we will in particular focus on periodic functions f whose dominant Fourier series coefficients are all associated with integer frequencies belonging to sets $S_1, \ldots, S_n \subset (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$, each of which is generated by an unknown degree $\leq d$ polynomial $P_j \in \mathbb{Z}[x]$. More specifically, we will assume that each set S_j is given by

$$S_j := \left\{ P_j(k) \ : \ k = 1, \ldots, B'_j \in \mathbb{N} \right\}, \tag{2}$$

where $0 < B'_j \leq B \ll N$ always holds for some support set cardinality upper bound $B \in \mathbb{N}$. Perhaps the simplest class of structured frequency sparse functions of this type are the *block frequency sparse functions* for which each $P_j(x) = x + a_j$ for some $a_j \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor - B] \cap \mathbb{Z}$.

Though our main results will concern the relatively general setting where our frequency support sets are given by (2), in what follows we will pay particular attention to the simpler class of block frequency sparse functions. Related block Fourier sparse structures appear in many signal processing contexts including, e.g., the reconstruction of multiband signals via blind sub-Nyquist sampling [11,32–35]. This class of block Fourier sparse functions also appears in related numerical methods for the rapid approximation of functions which exhibit sparsity with respect to other orthonormal basis functions. For example, one can rapidly approximate functions which are a sparse combination of high-degree Legendre polynomials by computing the DFT of samples from a related periodic function which is always guaranteed to be approximately block frequency sparse [21].

The importance of block frequency sparse functions has already led several authors to consider deterministic sub-linear time Fourier transforms for this case. Examples include several approaches which focus on the recovery of periodic functions whose frequency support is confined to just one block [4, 38, 39] or several blocks [8]. Herein we significantly generalize these first block frequency sparse recovery results by developing new deterministic SFT methods which enjoy recovery guarantees for all structured frequency sparse periodic functions f satisfying both (1) and (2). In particular, the methods proposed herein can rapidly recover block frequency sparse functions whose frequency support contains any given number of blocks.

1.3 Notation and Setup

We will always consider continuous 2π-periodic functions $f \colon [0, 2\pi] \to \mathbb{C}$ with $f(x) = \sum_{j=1}^{n} \sum_{\omega \in S_j} c_\omega(f) e^{i\omega x}$ where the unknown support sets $S_1, \ldots, S_n$ all satisfy (2). We will denote the Fourier series coefficients of any such f by $\mathbf{c}(f) = (c_\omega(f))_{\omega \in \mathbb{Z}}$ with

$$c_\omega(f) := \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-i\omega x} dx.$$

We will also consider perturbations of f by arbitrary 2π-periodic functions $\eta \in L^2([0, 2\pi])$ whose Fourier series coefficients $\mathbf{c}(\eta) \in \ell^1$ and also satisfy $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ for some $\varepsilon > 0$.

We will further denote by $\mathbf{c}(N) \in \mathbb{C}^N$ the restriction of the sequence $\mathbf{c}(f + \eta)$ to the

frequencies contained in $(-\lceil N/2\rceil, \lfloor N/2\rfloor] \cap \mathbb{Z}$, and by $\mathbf{c}(N, \mathbb{Z})$ the embedding of $\mathbf{c}(N)$ into $\mathbb{C}^{\mathbb{Z}}$:

$$(\mathbf{c}(N, \mathbb{Z}))_\omega = \begin{cases} c_\omega(f+\eta), & \omega \in (-\lceil N/2\rceil, \lfloor N/2\rfloor] \cap \mathbb{Z}, \\ 0, & \text{otherwise.} \end{cases}$$

A Fourier coefficient $c_\omega := c_\omega(f+\eta) \in \mathbb{C}$ will be called *significantly large* if $|c_\omega| > \varepsilon$. Similarly, a frequency $\omega \in \mathbb{Z}$ is *energetic* if its corresponding Fourier coefficient c_ω is significantly large.

For any vector $\mathbf{x} \in \mathbb{C}^{|I|}$ with index set I, and subset $R \subseteq I$, we define the vector $\mathbf{x}_R \in \mathbb{C}^{|I|}$ by

$$(\mathbf{x}_R)_i = \begin{cases} x_i, & \text{if } i \in R, \\ 0, & \text{otherwise} \end{cases}$$

for all $i \in I$. Furthermore, we denote by $\mathbf{0}_M \in \mathbb{C}^M$ the vector consisting of M zeroes and by $\mathbf{1}_M \in \mathbb{C}^M$ the vector consisting of M ones.

Finally, for any $s < |I|$ we will let the subset $R_s^{\text{opt}} \subset I$ be the, in lexicographical order, first s-element subset such that $|x_j| \geq |x_k|$ for all $j \in R_s^{\text{opt}}$ and $k \in I \backslash R_s^{\text{opt}}$. Thus, R_s^{opt} contains the indices of s entries of $\mathbf{x}$ with the largest magnitudes. While choosing s entries with the largest magnitude might not be unique, R_s^{opt} is unique. To simplify notation we set $\mathbf{x}_s^{\text{opt}} := \mathbf{x}_{R_s^{\text{opt}}}$. We will also say, e.g., that $(\mathbf{c}(N))_s^{\text{opt}} = \mathbf{c}_s^{\text{opt}}(N)$.

Throughout the remainder of this paper we will always consider samples to be taken from $f + \eta$ so that we are recovering a function which is potentially both non-sparse in Fourier domain and unstructured in its dominant frequency support. However, if, for example, the nonzero Fourier coefficients of f all satisfy $|c_\omega(f)| > 2\varepsilon$, then the structured frequency sparsity of f guarantees that

$$\{\omega \ : \ |c_\omega(f+\eta)| > \epsilon\} \cap R_{Bn}^{\text{opt}}(f+\eta) \subseteq S := \bigcup_{j=1}^{n} S_j.$$

It is exactly this type of consideration which will allow us to obtain near-optimal best Bn-term approximation guarantees for $f + \eta$ via our deterministic SFT methods below.

1.4 Results

As previously mentioned, we will confine our reconstruction results to the class of periodic functions, f, with structured frequency support satisfying both (1) and (2) above. Momentarily ignoring the structure of the support sets, S_j, given in (2) one can see that each such f is approximately Bn-sparse. As a result, it can be recovered in $B^2n^2 \log^{\mathcal{O}(1)} N$-time using the best deterministic SFTs for unstructured sparsity [23, 24]. Herein we obtain the following improved deterministic recovery result by taking the structure of the support sets (2) into account. It is a simplified corollary of Theorem 3.12 in §3.

Theorem 1.1
Let $f, \eta \in L^2([0, 2\pi])$ be as in §1.3. In addition, assume for simplicity that $\mathbf{c}_\omega(f + \eta) = 0$ for all $\omega \notin (-\lceil N/2\rceil, \lfloor N/2\rfloor] \cap \mathbb{Z}$, and that $B > n \log N$ where n is the number of polynomials of degree at most d which are evaluated at most B times to obtain the energetic frequencies in $S_1, \ldots, S_n$. In this case Algorithm 1 below is guaranteed to always

return a sparse N-length vector $\mathbf{x}_R$ of Fourier coefficient estimates that satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_2 + \sqrt{B}\left(\varepsilon + \frac{3}{d\sqrt{n}}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{2Bn}(N)\right\|_1\right) \quad (3)$$

when given access to

$$\mathcal{O}\left(\frac{Bd^2n^3\log^5 N}{\log B\log^2(dn)}\right)$$

samples from $f+\eta$ on $[0,2\pi]$. Furthermore, the runtime of Algorithm 1 is always

$$\mathcal{O}\left(\frac{Bd^2n^3\log^5 N}{\log^2(dn)}\right).$$

Note that algorithm mentioned in Theorem 1.1 will outperform existing deterministic SFTs with respect to runtime on functions with structured frequency support whenever $B \gg d^2 n \log N$. Most importantly, it does so while still maintaining a (slightly weakened) ℓ^2/ℓ^1 error guarantee (3) of the same type as the error guarantees of many compressive sensing methods [12].

Of course nothing comes for free. The error guarantee (3) is only really meaningful in the setting where the function with structured frequency support, f, dominates the arbitrary noise η in $f+\eta$. If, for example, $|c_\omega(f)| < 2\varepsilon$ for all $\omega \in S = \bigcup_{j=1}^n S_j$, then $f+\eta$ might not be approximated well by a function with structured frequency sparsity anymore. In such cases the runtime of Algorithm 1 in Theorem 1.1 will still be fast, but at the expense of the right hand side of (3) being relatively large (due to the ε-term). As a consequence, one can see that Theorem 1.1 only provides a meaningful computational improvement over standard deterministic SFTs in the case where, e.g., both $B \gg d^2 n \log N$ and

$$\varepsilon := \|\mathbf{c}(\eta)\|_\infty \lesssim \frac{1}{d\sqrt{n}}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_1 \leq \frac{1}{d\sqrt{n}}\sum_{\omega\notin S}|c_\omega(\eta)|$$

are true.

If one focuses on the more restrictive case of block frequency sparse functions, where all support sets S_j in (2) are generated by evaluating n linear, monic polynomials at B consecutive points, the methods developed herein also provide the following simplified result. It is a corollary of Theorem 4.1 in §4.

Theorem 1.2
Let $f, \eta \in L^2([0,2\pi])$ be as in §1.3 and let further f be block frequency sparse. In addition, assume for simplicity that $\mathbf{c}_\omega(f+\eta) = 0$ for all $\omega \notin (-\lceil N/2\rceil, \lfloor N/2\rfloor]\cap\mathbb{Z}$. In this case the variant of Algorithm 1 presented in §4 is guaranteed to always return a sparse N-length vector $\mathbf{x}_R$ of Fourier coefficient estimates that satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_1 \leq 4\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_1 + 2Bn\varepsilon$$

when given access to

$$\mathcal{O}\left(\frac{Bn^2\log^4 N}{\log^2 n}\right)$$

samples from $f + \eta$ *on* $[0, 2\pi]$. *Furthermore, the runtime of the algorithm is always*

$$\mathcal{O}\left(\frac{Bn^2 \log B \log^4 N}{\log^2 n}\right).$$

Inspecting Theorem 1.2 above one can see that it always provides a theoretical runtime improvement over existing $B^2n^2 \log^{\mathcal{O}(1)} N$-time methods for unstructured sparsity when applied to block frequency sparse functions. Moreover, a (slightly weakened) ℓ^1/ℓ^1 sparse approximation error guarantee is obtained. As above, we note that this result represents a significant improvement over existing techniques for this class of periodic functions as long as ε is sufficiently small (i.e., as long as $f + \eta$ is sufficiently well approximated by a block frequency sparse function).

1.5 Techniques and Overview

The deterministic SFT algorithms introduced in [23] implicitly construct compressive sensing matrices $\mathcal{M} \in \{0,1\}^{m \times N}$ with $m \ll N$ which have several useful properties, including (i) the restricted isometry property, (ii) they are the adjacency matrices of highly unbalanced expander graphs, and (iii) they are d-disjunct group testing matrices.[2] In addition to these properties, the matrices $\mathcal{M}$ also interact well with the Fourier basis in the following sense. Let $\mathbf{F} \in \mathbb{C}^{N \times N}$ be a discrete Fourier transform matrix. Then, the matrix product $\mathcal{M}\mathbf{F}$ is guaranteed to be highly sparse, with fewer than m columns containing nonzero entries.

It is precisely this collection of properties of $\mathcal{M}$ which ultimately allows for the development of the improved deterministic SFT algorithms presented in [24]. To get some intuition for how this works, one can consider the recovery of the approximately sparse Fourier coefficients $\mathbf{c}(N) \in \mathbb{C}^N$ using only the measurements $\mathcal{M}(\mathbf{c}(N)) \in \mathbb{C}^m$. The properties of $\mathcal{M}$ make it clear that such recovery is possible via, e.g., standard compressive sensing methods [12]. In fact, with more work one can show that the special properties of $\mathcal{M}$ allow for the recovery of $\mathbf{c}(N)$ in just $m \log^{\mathcal{O}(1)} m$-time, and without compromising the error guarantees one generally expects from compressive sensing algorithms such as Basis Pursuit. In addition, only a small number of samples from $f + \eta$ are required, since $\mathcal{M}\mathbf{F}$ is highly sparse, and

$$\mathcal{M}\mathbf{c}(N) = (\mathcal{M}\mathbf{F})\left(\mathbf{F}^{-1}\mathbf{c}(N)\right) = (\mathcal{M}\mathbf{F})\,\mathbf{x}$$

where $\mathbf{x}$ has $x_j = (f + \eta)\left(\frac{2\pi j}{N}\right)$. Thus, just a few entries of $\mathbf{x}$ have to be observed in order to obtain the necessary measurements $\mathcal{M}(\mathbf{c}(N))$.

In this paper we build on [23, 24] by augmenting the number theoretic constructions of the matrices $\mathcal{M}$ above in a way which allows us to benefit from structured frequency support while simultaneously preserving all the of properties of $\mathcal{M}$ needed in order to maintain extremely fast (i.e., sub-linear) runtimes. Intuitively, this is accomplished by augmenting a well chosen measurement matrix $\mathcal{M}$ from [24] with a set of several additional vectors $(\mathbf{u}_j)_{j=1}^M \subset \{0,1\}^N$ as follows. Let $\circ$ denote the Hadamard product and $\circledast$ the row-wise Hadamard product, where the first κ rows of $A \circledast B$ are given as the Hadamard product of all rows of A with the first row of B, the second κ rows as the Hadamard product of all rows of A with the second row of B and so forth. Below we will utilize a set of new measurement matrices $\mathcal{M} \circledast \mathbf{u}_1, \ldots, \mathcal{M} \circledast \mathbf{u}_M \in \{0,1\}^{m \times N}$.

[2] See [3] for additional details about these matrices and all of their remarkable properties.

Collectively, these new matrices are then shown to still allow all of the measurements $(\mathcal{M} \circledast \mathbf{u}_1)(\mathbf{c}(N)), \ldots, (\mathcal{M} \circledast \mathbf{u}_M)(\mathbf{c}(N)) \in \mathbb{C}^m$ to be computed using just a few samples from $f + \eta$. Furthermore, when $f + \eta$ is structured frequency sparse, it is shown that $\mathbf{u}_j \circ \mathbf{c}(N)$ will be guaranteed to be significantly more sparse than $\mathbf{c}(N)$ for most of the values of $j = 1, \ldots, M$. Hence, the deterministic SFT methods from [23, 24] will allow each such $\mathbf{u}_j \circ \mathbf{c}(N)$ to be recovered using the measurements

$$(\mathcal{M} \circledast \mathbf{u}_j)(\mathbf{c}(N)) = \mathcal{M}(\mathbf{u}_j \circ \mathbf{c}(N))$$

much faster than one can deterministically recover $\mathbf{c}(N)$ all at once using the same techniques.

Finally, the structure of the vectors $(\mathbf{u}_j)_{j=1}^M \subset \{0,1\}^N$ is then used to rapidly and accurately reconstruct $\mathbf{c}(N)$ from the set of its partial reconstructions of $(\mathbf{u}_j \circ \mathbf{c}(N))_{j=1}^M$. Here it becomes crucial to deal with the fact that the partial reconstructions of $\mathbf{u}_j \circ \mathbf{c}(N)$ are incorrect for some values of j. Thankfully, median arguments adapted from earlier SFT algorithms [13, 15] allow this to be handled easily by simply using enough vectors $(\mathbf{u}_j)_{j=1}^M$ in order to guarantee that the majority of the values of j provide good results. It then just remains to modify the reconstruction procedure from [24] in order to rapidly recover $\mathbf{c}(N)$ from the partial reconstructions of $(\mathbf{u}_j \circ \mathbf{c}(N))_{j=1}^M$.

The remainder of the paper is organized as follows: In §2 the vectors $(\mathbf{u}_j)_{j=1}^M$ discussed above are constructed, and it is proven that $\mathbf{u}_j \circ \mathbf{c}(N)$ will be approximately sparse for the majority of the $\mathbf{u}_j$ whenever $f + \eta$ exhibits sufficiently structured frequency support. Next, a deterministic reconstruction algorithm is developed for functions with structured frequency support in §3, and Theorem 1.1 is proven. These results are then improved for the simpler class of block frequency sparse signals in §4, and Theorem 1.2 is proven. Finally, the methods developed for block frequency sparse functions are empirically evaluated in §5. The paper then concludes with a short discussion of future work in §6.

2 Preliminaries

First, we formally define the notion of polynomially structured sparsity that was already mentioned in (2) in §1.

Definition 2.1 ($P(n,d,B)$-structured Sparsity)
Let $B, d, n, N \in \mathbb{N}$ such that $d < B < N$ and let $P_1, \ldots, P_n \in \mathbb{Z}[x]$ be non-constant polynomials of degree at most d with

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k,$$

where $a_{jk} \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ such that for all $j \in \{1, \ldots, n\}$ and $x \in \{1, \ldots, B\}$ we have $P_j(x) \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$. Define the n *support sets*

$$S_j := \{P_j(x) : x \in \{1, \ldots, B\}\}$$

and let $S := \bigcup_{j=1}^n S_j$. A 2π-periodic function $f \colon [0, 2\pi] \to \mathbb{C}$ is *P(n,d,B)-structured sparse* if it is of the form

$$f(x) = \sum_{\omega \in S} c_\omega(f) e^{i\omega x}$$

for some vector of Fourier coefficients $(c_\omega(f))_{\omega \in S} \in \mathbb{C}^{Bn}$.

This means that the at most Bn energetic frequencies of the function f are generated by evaluating n polynomials of degree at most d with integer coefficients at B points.

Our aim in this paper is to develop a sublinear-time Fourier algorithm for $P(n, d, B)$-structured sparse input functions, based on ideas introduced in [23] and [24]. One important concept for our method is that of a *good hashing prime*; a prime modulo which not all frequencies in a support set S_j are hashed to the same residue.

Definition 2.2
Let f be a $P(n, d, B)$-structured sparse function with support set $S = \bigcup_{j=1}^{n} S_j$ generated by some polynomials $P_1, \ldots, P_n$. Then a prime $p > B$ *hashes* a support set S_j *well* if

$$|\{\omega \mod p : \omega \in S_j\}| > 1.$$

Lemma 2.3
Let f be a $P(n, d, B)$-structured sparse function with support set $S = \bigcup_{j=1}^{n} S_j$ defined by some polynomials $P_1, \ldots, P_n$. Then a prime $p > B$ hashes a support set S_j with generating polynomial

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k$$

well if and only if there exists a non-constant coefficient a_{jk}, $k \neq 0$, with $p \nmid a_{jk}$.

Proof. Assume $p | a_{jk}$ for all $k \in \{1, \ldots, d\}$. Then we have for all $x \in \{1, \ldots, B\}$ that

$$P_j(x) = \sum_{k=0}^{d} a_{jk} x^k \equiv a_{j0} \mod p \quad \Rightarrow \quad |\{\omega \mod p : \omega \in S_j\}| = 1,$$

so p does not hash S_j well. If, on the other hand, p does not hash S_j well, then

$$|\{\omega \mod p : \omega \in S_j\}| = 1 \quad \Rightarrow \quad P_j(y) \equiv P_j(z) \mod p \quad \forall y, z \in \{1, \ldots, B\}.$$

This means that for fixed $y \in \{1, \ldots, B\}$ the polynomial

$$Q(x) := P_j(x) - P_j(y) = \sum_{k=0}^{d} a_{jk} x^k - P_j(y)$$

of degree d has $B > d$ zeroes modulo p. Thus Q is the zero polynomial modulo p, and

$$p | (a_{j0} - P_j(y)) \quad \text{and} \quad p | a_{jk} \quad \forall j \in \{1, \ldots, d\}.$$

□

For a good hashing prime and a $P(n, d, B)$-structured sparse function we can bound the number of frequencies that are hashed to the same residue.

Lemma 2.4
Let f be a $P(n, d, B)$-structured sparse function with support set $S = \bigcup_{j=1}^{n} S_j$ defined by some polynomials $P_1, \ldots, P_n$. If a support set S_j is hashed well by a prime $p > B$, then

(i) P_j is not constant modulo p and

(ii) $|\{\omega \equiv \nu \mod p : \omega \in S_j\}| \leq d$ for all residues $\nu \in \{0, \ldots, p-1\}$.

Proof. It is clear that P_j is not constant modulo p if $|\{\omega \mod p : \omega \in S_j\}| > 1$. Assume now that $|\{\omega \equiv \nu \mod p : \omega \in S_j\}| > d$ for some $\nu \in \{0, \ldots, p-1\}$. Since all elements of S_j are generated by evaluating P_j at B points, we find for a $y \in \{1, \ldots, B\}$ with $P_j(y) \equiv \nu \mod p$ that

$$P_j(y) \equiv P_j(z) \mod p$$

for d distinct choices of $z \in \{1, \ldots, B\} \backslash \{y\}$. Then the polynomial $Q(x) := P_j(x) - P_j(y)$ has at least $d+1$ zeroes modulo p, which is a contradiction, so (ii) holds. □

Let us now assume that there exists a prime $p > B$ that hashes all support sets $S_1, \ldots, S_n$ of a $P(n, d, B)$-structured sparse function well. Then the restriction of any S_j to the frequencies congruent to ν modulo p is at most d-sparse for all residues $\nu \in \{0, \ldots, p-1\}$. Consequently, the restriction of S to these frequencies is at most dn-sparse.

In our setting of $P(n, d, B)$-structured sparse input functions we want to apply the SFT algorithm in [24] (Algorithm 3) to the restrictions of the function to frequencies congruent to ν modulo u for all residues ν, where u is a prime that hashes all support sets well, since these restrictions are at most dn-sparse.

In general, finding a single well-hashing prime u for all support sets is not possible without further information on the generating polynomials. However, we can use the observations presented in the remainder of §2 to find M primes such that the majority of them hashes all support sets well. As these methods, as well as Algorithm 3 in [24], rely heavily on the Chinese Remainder Theorem, we state it here as a reminder (see [28]).

Theorem 2.5 (Chinese Remainder Theorem (CRT))
Let $n_1, \ldots, n_m$ be pairwise relatively prime integers and $N \leq \prod_{j=1}^{m} n_j$. Then the system of simultaneous congruencies $x \equiv a_1 \mod n_1, \ldots, x \equiv a_m \mod n_m$ has a unique solution modulo N.

Definition 2.6 (Enumeration of the Natural Primes)
For $j \in \mathbb{N}$ denote by p_j the j-th natural prime number. Additionally, let $p_0 = 1$. Then,

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

In order to find primes for which the restriction of the frequencies to any residue is sparse, we first need to define the notion of separation.

Definition 2.7 (Separation)
Let $k \in \mathbb{N}$ and $\omega_1, \ldots, \omega_k \in \mathbb{Z}$. An integer $n \in \mathbb{N}$ is said to *separate* $\omega_1, \ldots, \omega_k$ if

$$\omega_j \mod n \neq \omega_l \mod n \quad \forall j, l \in \{1, \ldots, k\},\ j \neq l.$$

The following result about separating primes has been shown in [24].

Lemma 2.8
Let $E \in \mathbb{N}$ and $u_1 := p_r$ for some $r \in \mathbb{N}$. Set $M = 2 \cdot E \cdot \lfloor \log_{u_1} N \rfloor + 1$. Choose $M-1$ further primes with $u_1 < \cdots < u_M$ and let $T \subset (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ with $|T| \leq E$. Then more than $\frac{M}{2}$ of the u_m separate every $x \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ from all $t \in T \backslash \{x\}$.

In the next lemma we prove that, for a suitable M, it suffices to find M primes such that more than half of them separate the leading coefficients of the frequency generating polynomials from 0 at the same time, in order to guarantee that more than half of them hash all support sets well.

Lemma 2.9
Let f be $P(n,d,B)$-structured sparse with support set $S = \bigcup_{j=1}^{n} S_j$ defined by some polynomials $P_1, \ldots, P_n$, and set $E = n+1$. Let M primes $B < u_1 < \cdots < u_M$ be given as in Lemma 2.8. Then more than $\frac{M}{2}$ of the u_m hash all n support sets $S_1, \ldots, S_n$ well.

Proof. Let T be the set consisting of the distinct leading polynomial coefficients,

$$T := \left\{ a_{j,\deg(P_j)} : P_j(x) = \sum_{k=0}^{\deg(P_j)} a_{jk}x^k,\ j \in \{1,\ldots,n\} \right\}.$$

Then $a_{j,\deg(P_j)} \neq 0$ for all polynomials and, since $|T \cup \{0\}| \leq E$, by Lemma 2.8 more than $\frac{M}{2}$ of the u_m separate every element of $(-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ from all other elements of $T \cup \{0\}$, i.e., from all distinct leading polynomial coefficients and from 0.

Let p be such a prime. Assume that there exists a support set S_j that is not well hashed by p, so that we have

$$\{\omega \bmod p : \omega \in S_j\} = \{\nu\}$$

for some residue $\nu \in \{0, \ldots, p-1\}$. Then the polynomial P_j that generates S_j satisfies

$$P_j(x) - \nu \equiv 0 \bmod p \quad \forall x \in \{1, \ldots, B\}.$$

Consider now the polynomial $Q(x) := P_j(x) - \nu$ modulo p. It is a polynomial of degree at most d with $B > d$ zeroes, so it has to be the zero polynomial modulo p, meaning that

$$p | a_{jk} \quad \forall k \in \{1, \ldots, d\} \quad \text{and} \quad p | (a_{j0} - \nu).$$

Since p separates $a_{j,\deg(P_j)}$ from 0 and the other leading coefficients, we find that

$$a_{j,\deg(P_j)} \equiv 0 \bmod p \quad \text{and} \quad a_{j,\deg(P_j)} \not\equiv t \bmod p \quad \forall t \in (T \cup \{0\}) \backslash \{a_{j,\deg(P_j)}\}.$$

This is only possible if $a_{j,\deg(P_j)} = 0$, which is a contradiction. Thus we obtain that

$$|\{\omega \bmod p : \omega \in S_j\}| > 1,$$

so p hashes all S_j well. Consequently, all of the more than $\frac{M}{2}$ primes $u_1, \ldots, u_M$ that separate the leading coefficients from one another and from 0 hash all support sets well. □

These lemmas imply that applying Algorithm 3 in [24] to the restrictions of the input function to frequencies congruent to ν modulo M primes as defined in Lemma 2.8 yields the correct frequencies and Fourier coefficients in the majority of the cases, since the algorithm works well as long as the input function is sparse enough.

3 Algorithm for Polynomially Structured Sparse Functions

Before we can begin to develop our algorithm for polynomially structured sparse functions, we need to develop the notation necessary in order to apply Algorithm 3 in [24] to the frequency restrictions.

3.1 Measurement Matrices

Definition 3.1 (Notation)
Let $f\colon [0,2\pi] \to \mathbb{C}$ be $P(n,d,B)$-structured sparse with bandwidth N. Let $B < u_1 < \cdots < u_M$ be prime and $s_1 < \cdots < s_K$ pairwise relatively prime natural numbers such that there exist L natural numbers $t_1 < \cdots < t_L < s_1$ which satisfy that the set

$$\{t_1, \ldots, t_L, s_1, \ldots, s_K, u_1, \ldots, u_M\}$$

is pairwise relatively prime and that

$$\prod_{l=1}^{L} t_l \geq \frac{N}{s_1 u_1}.$$

We set $\kappa := \sum_{k=1}^{K} s_k$, $\lambda := 1 + \sum_{l=1}^{L} t_l$ and $\mu := \sum_{m=1}^{M} u_m$. Further, we define

$$q := \operatorname{lcm}(N, s_1, \ldots, s_K, t_1, \ldots, t_L, u_1, \ldots, u_M).$$

From now on we always assume that occurring natural numbers $q, s_1, \ldots, s_K, t_1, \ldots, t_L$, $u_1, \ldots, u_M$ comply with Definition 3.1.

Definition 3.2
For a given 2π-periodic function f and $m \in \mathbb{N}$ define the sample vector $\mathbf{A}_m \in \mathbb{C}^m$ by

$$\mathbf{A}_m(j) := f\left(\frac{2\pi j}{m}\right) \quad \forall j \in \{0, \ldots, m-1\}.$$

Definition 3.3 (Discrete Fourier Transform)
For $m \in \mathbb{N}$ we denote the discrete Fourier transform of a vector $\mathbf{x} \in \mathbb{C}^m$ by

$$\widehat{\mathbf{x}} = \mathbf{F}_m \mathbf{x},$$

where $\mathbf{F}_m := \frac{1}{m}\left(\omega_m^{jk}\right)_{j,k=0}^{m-1} \in \mathbb{C}^{m\times m}$ is the m-th Fourier matrix and $\omega_m := e^{\frac{-2\pi i}{m}}$ is the m-th primitive root of unity.

In order to apply Algorithm 3 in [24] to the restrictions to frequencies that are congruent to ν modulo u_m for all residues $\nu \in \{0, \ldots, u_m - 1\}$ and all $m \in \{1, \ldots, M\}$, we need to transform the vector $\widehat{\mathbf{A}_q}$ into a matrix with sparse columns whose entries correspond to the frequencies that are congruent to ν modulo u_m. For this purpose and also for later use we recall the definition of the row-wise Hadamard tensor product.

Definition 3.4 (Row-wise Hadamard Product)
Let $\mathbf{A} \in \mathbb{C}^{\kappa\times m}$, $\mathbf{B} \in \mathbb{C}^{\lambda\times m}$. Then the row-wise Hadamard product $\mathbf{A} \circledast \mathbf{B} \in \mathbb{C}^{(\kappa\cdot\lambda)\times m}$ is

given by

$$(\mathbf{A} \circledast \mathbf{B})_{jl} = \mathbf{A}_{j \bmod \kappa, l} \cdot \mathbf{B}_{\frac{j-(j \bmod \kappa)}{\kappa}, l}, \quad j \in \{0, \dots, \kappa\lambda - 1\},\, l \in \{0, \dots, m-1\},$$

i.e., the first κ rows are given as the Hadamard product of all rows of $\mathbf{A}$ with the first row of $\mathbf{B}$, the second κ rows as the Hadamard product of all rows of $\mathbf{A}$ with the second row of $\mathbf{B}$ and so forth.

Lemma 3.5
Let $\mathbf{A} \in \mathbb{C}^{\kappa \times m}$, $\mathbf{B} \in \mathbb{C}^{\lambda \times m}$. Then every row of $\mathbf{A} \circledast \mathbf{B}$ is given as the row tensor product of a row of $\mathbf{A}$ with a row of $\mathbf{B}$.

Definition 3.6 (Measurement Matrices I)
For $t_1, \dots, t_L, s_1, \dots, s_K, u_1, \dots, u_M$ from Definition 3.1 we construct a special $\mu \times N$ measurement matrix $\mathcal{M}_{u_1,M}$, analogously to the measurement matrix concept used in [24]. The matrix consists of rows of ones and zeroes, where an entry of a row is one if and only if its column index is congruent to a certain residue modulo u_m. Let $m \in \{1, \dots, M\}$ and $\nu \in \{0, \dots, u_m - 1\}$ be a fixed residue modulo u_m. Then we define the row $\mathbf{r}_{u_m,\nu}$ by

$$(\mathbf{r}_{u_m,\nu})_j := \delta\left((j-\nu) \bmod u_m\right) = \begin{cases} 1, & \text{if } j \equiv \nu \bmod u_m, \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

and set

$$\mathcal{M}_{u_1,M} := \begin{pmatrix} \mathbf{r}_{u_1,0} \\ \vdots \\ \mathbf{r}_{u_1,u_1-1} \\ \mathbf{r}_{u_2,0} \\ \vdots \\ \mathbf{r}_{u_M,u_M-1} \end{pmatrix} = \begin{pmatrix} I_{u_1} & I_{u_1} & \dots \\ I_{u_2} & I_{u_2} & \dots \\ \vdots & \vdots & \vdots \\ I_{u_M} & I_{u_M} & \dots \end{pmatrix}.$$

We define the extension $\mathcal{H}_M$ of $\mathcal{M}_{u_1,M}$ to a $\mu \times q$ matrix by extending all rows $\mathbf{r}_{u_m,\nu}$ to columns indexed by $j \in \{0, \dots, q-1\}$, as given by (4). Further, as in [24], we define the $\kappa \times N$ matrix $\mathcal{M}_{s_1,K}$ and the $(\lambda - 1) \times N$ matrix $\mathcal{M}_{t_1,L}$, consisting of the rows corresponding to the possible residues modulo all the s_k and t_l, respectively,

$$\mathcal{M}_{s_1,K} := \begin{pmatrix} \mathbf{r}_{s_1,0} \\ \vdots \\ \mathbf{r}_{s_K,s_K-1} \end{pmatrix} \quad \text{and} \quad \mathcal{M}_{t_1,L} := \begin{pmatrix} \mathbf{r}_{t_1,0} \\ \vdots \\ \mathbf{r}_{t_L,t_L-1} \end{pmatrix}.$$

We set $\mathcal{N}_{t_1,L}$ to be the $\lambda \times N$ matrix whose first row contains only ones and whose other rows are given by $\mathcal{M}_{t_1,L}$,

$$\mathcal{N}_{t_1,L} := \begin{pmatrix} \mathbf{1}_N \\ \mathcal{M}_{t_1,L} \end{pmatrix},$$

and define the $(\kappa \cdot \lambda) \times N$ row-wise Hadamard product $\mathcal{R}_{L,K} := \mathcal{M}_{s_1,K} \circledast \mathcal{N}_{t_1,L}$ of $\mathcal{M}_{s_1,K}$ and $\mathcal{N}_{t_1,L}$ and its extension $\mathcal{G}_{L,K}$ to a $(\kappa \cdot \lambda) \times q$ matrix. Because all s_k, t_l and u_m thus have to divide q, we set $q = \operatorname{lcm}(N, s_1, \dots, s_K, t_1, \dots, t_L)$ as the length of the sample vector.

Remark 3.7 (Restriction of $\widehat{\mathbf{A}_q}$)

If we compute the row-wise Hadamard product of $\mathcal{H}_M$ with $(\widehat{\mathbf{A}_q})^T \in \mathbb{C}^{1\times q}$, every row of $\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T$ is, by Lemma 3.5, given as the row-wise Hadamard product of a row of $\mathcal{H}_M$ and $(\widehat{\mathbf{A}_q})^T$. Thus every column $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T \in \mathbb{C}^{q\times\mu}$ corresponds to a residue ν modulo a prime u_m. This column only contains nonzero entries at frequencies that are congruent to ν modulo u_m,

$$\begin{aligned}\boldsymbol{\rho}^T_{u_m,\nu}(j) &:= \left(\mathbf{r}_{u_m,\nu} \circledast (\widehat{\mathbf{A}_q})^T\right)^T_j = (\mathbf{r}_{u_m,\nu})^T_j \cdot (\widehat{\mathbf{A}_q})_j \\ &= \delta\left((j-\nu) \bmod u_m\right) \cdot \widehat{\mathbf{A}_q}(j) = \begin{cases}\widehat{\mathbf{A}_q}(j), & j \equiv \nu \bmod u_m, \\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

This means that the column $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ is the restriction of $\widehat{\mathbf{A}_q}$ to frequencies congruent to ν modulo u_m, which is at most dn-sparse for a good hashing prime u_m. Thus for more than $M/2$ of the u_m we can apply Algorithm 3 in [24] with sparsity dn column by column. ◊

3.2 Required Technical Background

In the following we give a short description of Algorithm 3 in [24] and summarize some of the results proven therein. Said SFT algorithm reconstructs the energetic frequencies and the corresponding Fourier coefficients of a sparse input function $f\colon [0, 2\pi] \to \mathbb{C}$ with bandwidth N from the Fourier transforms of vectors consisting of $s_k t_l \ll N$ equispaced samples of f, where s_k and t_l are small primes depending on the bandwidth and sparsity of the function. The energetic frequencies are reconstructed from their residues modulo s_k and $t_1, \ldots, t_L$ with the help of the CRT, which is why the primes have to satisfy

$$\prod_{l=1}^{L-1} t_l < \frac{N}{s_1} \leq \prod_{l=1}^{L} t_l.$$

For a general d-sparse input function the method introduced in [23, 24] is not guaranteed to work for any prime s_k. However, setting $K = 8d\lfloor \log_{s_1} N \rfloor + 1$ and choosing $s_1, \ldots, s_K$ as the K smallest primes greater than d and t_L, for more than $K/2$ of them all energetic frequencies are correctly reconstructed from their residues. These can be found by comparing the entries of $\widehat{\mathbf{A}_{s_k}}$ and $\widehat{\mathbf{A}_{s_k t_l}}$ that correspond to the same frequency for all l. The coefficient estimates are then obtained by taking the medians over the K coefficient estimates found for the s_k. The following remark summarizes the main results for Algorithm 3 in [24] that are relevant for this paper.

Remark 3.8
Let $f\colon [0, 2\pi] \to \mathbb{C}$ and $d < N \in \mathbb{N}$.

(i) (Lemma 5 in [24]) By the CRT, every row of $\mathcal{G}_{L,K}$ is of the form

$$\left(\mathbf{r}_{s_k t_l, h}\right)_j = \left(\mathbf{r}_{s_k, h \bmod s_k} \circledast \mathbf{r}_{t_l, h \bmod t_l}\right)_j = \begin{cases}1, & \text{if } j \equiv h \bmod s_k t_l \\ 0, & \text{otherwise}\end{cases},$$

for $h \in \{0, \ldots, s_k t_l - 1\}$, $k \in \{1, \ldots, K\}$ and $l \in \{0, \ldots, L\}$, where $t_0 := 1$ in order to have the same notation also for rows of the form $\mathbf{r}_{s_k, h \bmod s_k} \circledast \mathbf{1}_N$.

(ii) (Lemma 6 in [24]) If $\omega \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ is such that

$$|c_\omega| > 4 \cdot \left(\frac{1}{2d} \left\| \mathbf{c}(N) - \mathbf{c}_{2d}^{\mathrm{opt}}(N) \right\|_1 + \| \mathbf{c}(f) - \mathbf{c}(N, \mathbb{Z}) \|_1 \right),$$

then ω will be reconstructed more than $\frac{K}{2}$ times.

(iii) (Proof of Theorem 7 in [24]) If ω is reconstructed more than $\frac{K}{2}$ times, then

$$|x_\omega - c_\omega| \leq \sqrt{2} \left(\frac{1}{2d} \left\| \mathbf{c}(N) - \mathbf{c}_{2d}^{\mathrm{opt}}(N) \right\|_1 + \| \mathbf{c}(f) - \mathbf{c}(N, \mathbb{Z}) \|_1 \right).$$

(iv) (Theorem 7 in [24]) Algorithm 3 in [24] will output an $\mathbf{x}_R \in \mathbb{C}^N$ satisfying

$$\begin{aligned} &\| \mathbf{c}(N) - \mathbf{x}_R \|_2 \\ &\leq \left\| \mathbf{c}(N) - \mathbf{c}_d^{\mathrm{opt}}(N) \right\|_2 + \frac{11}{\sqrt{d}} \left\| \mathbf{c}(N) - \mathbf{c}_{2d}^{\mathrm{opt}}(N) \right\|_1 + 22\sqrt{d} \, \| \mathbf{c}(f) - \mathbf{c}(N, \mathbb{Z}) \|_1 \end{aligned}$$

in a runtime of

$$\mathcal{O}\left(\frac{d^2 \log^2 N \log(d \log N) \log^2 \frac{N}{d}}{\log^2 d \log \log \frac{N}{d}} \right).$$

$\Diamond$

3.3 Application to Polynomially Structured Sparse Functions

We can now apply Algorithm 3 in [24] with sparsity dn to the columns $\boldsymbol{\rho}_{u_m,\nu}^T$ of $(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$. Recall that $\boldsymbol{\rho}_{u_m,\nu}^T$ is only guaranteed to be at most dn-sparse if u_m hashes all support sets $S_1, \ldots, S_n$ well. This means that only for columns corresponding to those primes the algorithm will return all energetic frequencies and good estimates for their Fourier coefficients. Hence we have to apply the algorithm to every single column of $(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ and choose those frequencies and coefficient estimates that appear for the more than $\frac{M}{2}$ well-hashing u_m.

In Algorithm 3 in [24], estimates for the Fourier coefficients c_ω of the input function f are calculated from certain entries of $\mathcal{G}_{L,K} \cdot \widehat{\mathbf{A}_q}$. These entries can be obtained in a fast way from f by computing DFTs of the vectors $\mathbf{A}_{s_k t_l}$.

Remark 3.9
For polynomially structured sparse input functions we now have to show that the entries of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ can also be calculated fast. As we want to use the residues modulo the u_m for the reconstruction as well, an idea similar to the one from [24] leads to DFTs of the $s_k t_l u_m$-length sample vectors $\mathbf{A}_{s_k t_l u_m}$ from Definition 3.1. Consider an entry of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T \in \mathbb{C}^{\kappa\lambda \times \mu}$ that is given as the product of a row of $\mathcal{G}_{L,K}$, by Remark 3.8 of the form $\mathbf{r}_{s_k t_l, h}$ for a residue h modulo $s_k t_l$, with a column $\boldsymbol{\rho}_{u_m,\nu}^T$ of

$(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ for a residue ν modulo u_m,

$$\begin{aligned}\mathbf{r}_{s_k t_l,h} \cdot \boldsymbol{\rho}_{u_m,\nu}^T &= \sum_{j=0}^{q-1} \mathbf{r}_{s_k t_l,h}(j) \boldsymbol{\rho}_{u_m,\nu}^T(j) \\ &= \sum_{j=0}^{q-1} \delta\left((j-h) \mod s_k t_l\right) \delta\left((j-\nu) \mod u_m\right) \cdot \widehat{\mathbf{A}_q}(j). \end{aligned} \tag{5}$$

As there can only be nonzero summands in (5) if $j \equiv h \mod s_k t_l$ and $j \equiv \nu \mod u_m$, we find with the CRT that j has to be of the form

$$j = \tau + j' s_k t_l u_m, \quad j' \in \left\{0, \ldots, \frac{q}{s_k t_l u_m} - 1\right\}.$$

Then,

$$\begin{aligned}\mathbf{r}_{s_k t_l,h} \cdot \boldsymbol{\rho}_{u_m,\nu}^T &= \sum_{j'=0}^{\frac{q}{s_k t_l u_m}-1} \widehat{\mathbf{A}_q}(\tau + j' \cdot s_k t_l u_m) \\ &= \sum_{j'=0}^{\frac{q}{s_k t_l u_m}-1} \frac{1}{q} \sum_{b=0}^{q-1} \mathbf{A}_q(b) e^{\frac{-2\pi i b(\tau + j' s_k t_l u_m)}{q}} \\ &= \sum_{b=0}^{q-1} \frac{1}{q} \mathbf{A}_q(b) e^{\frac{-2\pi i b \tau}{q}} \sum_{j'=0}^{\frac{q}{s_k t_l u_m}-1} e^{\frac{-2\pi i b j' s_k t_l u_m}{q}} \\ &= \sum_{b=0}^{q-1} \frac{1}{s_k t_l u_m} f\left(\frac{2\pi b}{q}\right) e^{\frac{-2\pi i b \tau}{q}} \cdot \delta\left(b \mod \frac{q}{s_k t_l u_m}\right) \\ &= \sum_{b'=0}^{s_k t_l u_m - 1} \frac{1}{s_k t_l u_m} f\left(\frac{2\pi b' \frac{q}{s_k t_l u_m}}{q}\right) e^{\frac{-2\pi i \tau b' \frac{q}{s_k t_l u_m}}{q}} \\ &= \widehat{\mathbf{A}_{s_k t_l u_m}}(\tau). \end{aligned}$$

By Bézout's identity, $1 = \gcd(s_k t_l, u_m) = v \cdot s_k t_l + w \cdot u_m$ for some $v, w \in \mathbb{Z}$, and we obtain that τ satisfies

$$\tau = ((h-\nu) w \mod s_k t_l) \cdot u_m + \nu \in \{0, \ldots, s_k t_l u_m - 1\}. \tag{6}$$

Thus we find that, in the column for the residue ν modulo u_m, for fixed $s_k t_l$ only the $s_k t_l$ different values $\widehat{\mathbf{A}_{s_k t_l u_m}}(\tau)$ with τ depending on h as in (6) are contained. Hence the column of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ corresponding to ν modulo u_m is of the form

$$\left(\widehat{\mathbf{B}_{s_1}^{m,\nu}}^T, \widehat{\mathbf{B}_{s_1 t_1}^{m,\nu}}^T, \ldots, \widehat{\mathbf{B}_{s_1 t_L}^{m,\nu}}^T, \widehat{\mathbf{B}_{s_2}^{m,\nu}}^T, \ldots, \widehat{\mathbf{B}_{s_K t_L}^{m,\nu}}^T\right)^T,$$

where

$$\widehat{\mathbf{B}_{s_k t_l}^{m,\nu}}(j) := \widehat{\mathbf{A}_{s_k t_l u_m}}\left((j-\nu) w \mod s_k t_l) \cdot u_m + \nu\right) \quad \forall j \in \{0, \ldots, s_k t_l - 1\}. \tag{7}$$

The entries of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ can be calculated in a fast way, using KLM DFTs of vectors of $s_k t_l u_m$ equispaced samples with runtime $\mathcal{O}(s_k t_l u_m \cdot \log(s_k t_l u_m))$ for all k, l, m.

How exactly do we apply Algorithm 3 in [24] to the columns of $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$? Until now we considered a fixed residue h modulo $s_k t_l$ and a fixed residue ν modulo u_m. However, in line 7 of that algorithm we fix the residue h' modulo s_k of a frequency and find its residues modulo the $s_k t_l$ in line 9. In the setting of polynomially structured sparse functions this means that for a frequency ω with residue ν modulo u_m and residue h' modulo s_k we have to find the corresponding residue modulo $s_k t_l u_m$ for every l. Then

$$\tau' := \omega \mod s_k u_m = ((h' - \nu) w \mod s_k) \cdot u_m + \nu$$

holds for the residue of ω modulo $s_k u_m$, where Bézout's identity implies that

$$1 = \gcd(s_k, u_m) = v' \cdot s_k + w' \cdot u_m$$

for some $v', w' \in \mathbb{Z}$. Then the residue of ω modulo $s_k t_l u_m$ is of the form

$$\omega \mod s_k t_l u_m = \tau' + b_{\min} \cdot s_k u_m$$

for a $b_{\min} \in \{0, \ldots, t_l - 1\}$, which is given as

$$b_{\min} := \operatorname*{argmin}_{b \in \{0,\ldots,t_l-1\}} \left| \widehat{\mathbf{A}_{s_k u_m}}(\tau') - \widehat{\mathbf{A}_{s_k t_l u_m}}(\tau' + b \cdot s_k u_m) \right|. \tag{8}$$

Finally, the residue of ω modulo t_l is

$$a_l := \omega \mod t_l = (\tau' + b_{\min} \cdot s_k u_m) \mod t_l,$$

and ω can be reconstructed from its residues $\omega \equiv \tau' \mod s_k u_m$, $\omega \equiv a_1 \mod t_1$,…,$\omega \equiv a_L \mod t_L$. Recall the notion of the $\widehat{\mathbf{B}^{(m,\nu)}_{s_k t_l}}$ introduced in (7). These vectors are defined such that if $\omega \equiv \nu \mod u_m$ and $\omega \equiv h \mod s_k t_l$, we have

$$\widehat{\mathbf{B}^{m,\nu}_{s_k t_l}}(\omega \mod s_k t_l) = \widehat{\mathbf{A}_{s_k t_l u_m}}(\omega \mod s_k t_l u_m).$$

To use this notation, we take the residues modulo $s_k t_l u_m$ again modulo $s_k t_l$, and obtain the following,

$$\begin{aligned}
b_{\min} &= \operatorname*{argmin}_{b \in \{0,\ldots,t_l-1\}} \left| \widehat{\mathbf{A}_{s_k u_m}}(\tau') - \widehat{\mathbf{A}_{s_k t_l u_m}}(\tau' + b \cdot s_k u_m) \right| \\
&= \operatorname*{argmin}_{b \in \{0,\ldots,t_l-1\}} \left| \widehat{\mathbf{B}^{m,\nu}_{s_k}}(h') - \widehat{\mathbf{B}^{m,\nu}_{s_k t_l}}((\tau' + b \cdot s_k u_m) \mod s_k t_l) \right| \\
&= \operatorname*{argmin}_{b \in \{0,\ldots,t_l-1\}} \left| \left(\mathcal{E}_K \cdot \widehat{\mathbf{A}_q} \right)_{\mathbf{r}_{s_k,h'}, \boldsymbol{\rho}^T_{m,\nu}} - \left(\mathcal{G}_{L,K} \cdot \widehat{\mathbf{A}_q} \right)_{\mathbf{r}_{s_k t_l, (\tau' + b s_k u_m) \bmod s_k t_l}, \boldsymbol{\rho}^T_{m,\nu}} \right|.
\end{aligned}$$

$\Diamond$

Algorithm 1 presents itself as a summary of the preceding considerations.

Algorithm 1 Fourier Approximation

Input: Function $f+\eta$, $K = 8dn\lfloor \log_{s_1} \frac{N}{u_1} \rfloor + 1$, $M = 2(n+1)\cdot\lfloor \log_{u_1} N \rfloor + 1$ and pairwise relatively prime $s_1 < \cdots < s_K, t_1 < \cdots < t_L, B < u_1 < \cdots < u_M$ with $t_L < s_1$, u_m prime and $\prod_{l=1}^{L} t_l \geq \frac{N}{s_1 u_1}$.

Output: $R, \mathbf{x}_R$, where R contains the nB frequencies ω with greatest magnitude coefficient estimates $x_R(\omega)$.

Initialize $R = \emptyset$, $\mathbf{x}_R = \mathbf{0}_N$, $q = \mathrm{lcm}(N, s_1, \ldots, s_K, t_1, \ldots, t_L, u_1, \ldots, u_M)$

$\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T \leftarrow \left(\left(\widehat{\mathbf{B}_{s_1}^{m,\nu}}^T, \widehat{\mathbf{B}_{s_1 t_1}^{m,\nu}}^T, \ldots, \widehat{\mathbf{B}_{s_K t_L}^{m,\nu}}^T \right)^T \right)_{m=1,\nu=0}^{M,u_m-1}$

$\mathcal{E}_K \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T \leftarrow \left(\left(\widehat{\mathbf{B}_{s_1}^{m,\nu}}^T, \ldots, \widehat{\mathbf{B}_{s_K}^{m,\nu}}^T \right) \right)_{m=1,\nu=0}^{M,u_m-1}$

for m from 1 to M **do**

  for ν from 0 to $u_m - 1$ **do**

    $(R^{(m,\nu)}, \mathbf{x}^{(m,\nu)}) \leftarrow 2dn$ frequencies with largest magnitude coefficient estimates returned by Algorithm 3 in [24] applied to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T_{\boldsymbol{\rho}_{m,\nu}^T}$ and $\mathcal{E}_K \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T_{\boldsymbol{\rho}_{m,\nu}^T}$ with sparsity dn.

  end for

end for

for each $\omega \in \bigcup_{m=1}^{M} \bigcup_{\nu=0}^{u_m-1} R^{(m,\nu)}$ found more than $\frac{M}{2}$ times **do**

  $\mathrm{Re}(x_\omega) = \underset{\substack{\nu=0,\ldots,u_m-1 \\ m=1,\ldots,M}}{\mathrm{median}} \left\{ \mathrm{Re}\left(x_{\tilde{\omega}}^{(m,\nu)} \right) : \tilde{\omega} = \omega, \tilde{\omega} \in R^{(m,\nu)} \right\}$

  $\mathrm{Im}(x_\omega) = \underset{\substack{\nu=0,\ldots,u_m-1 \\ m=1,\ldots,M}}{\mathrm{median}} \left\{ \mathrm{Im}\left(x_{\tilde{\omega}}^{(m,\nu)} \right) : \tilde{\omega} = \omega, \tilde{\omega} \in R^{(m,\nu)} \right\}$

end for

Sort the coefficients by magnitude s.t. $|x_{\omega_1}| \geq |x_{\omega_2}| \geq \cdots$

Output: $R = \{\omega_1, \ldots, \omega_{nB}\}, \mathbf{x}_R$

3.4 Results for Polynomially Structured Sparse Functions

In order to obtain bounds on the accuracy and runtime of our algorithm, we can utilize some of the results developed in [24], at least for the more than $\frac{M}{2}$ primes u_m that hash all support sets $S_1, \dots, S_n$ well, i.e., the primes where the columns $\boldsymbol{\rho}^T_{u_m,\nu}$ of $(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$ are guaranteed to be at most dn-sparse.

Analogously to our previous notation we denote by $\mathbf{c}(N, u_m, \nu)$, $\mathbf{c}(N, \mathbb{Z}, u_m, \nu)$ and $\mathbf{c}(u_m, \nu)$ the restrictions of $\mathbf{c}(N)$, $\mathbf{c}(N, \mathbb{Z})$ and $\mathbf{c}(f + \eta)$, respectively, to the frequencies congruent to ν modulo u_m. Further, recall that $\mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu)$ is the optimal $2dn$-term representation of $\mathbf{c}(N, u_m, \nu)$.

The following lemma guarantees that all significantly enough frequencies will be found and their Fourier coefficients estimated well.

Lemma 3.10
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be $P(n, d, B)$-structured sparse with noise η such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let u_1 be a prime and s_1, t_1 natural numbers such that with $K = 8dn\lfloor \log_{s_1} \frac{N}{u_1} \rfloor + 1$ and $M = 2(n+1)\lfloor \log_{u_1} N \rfloor + 1$ we have that $B < u_1 < \cdots < u_M$ are primes, $t_1 < \cdots < t_L < s_1 < \cdots < s_K$, and $u_1, \dots, u_M, s_1, \dots, s_K, t_1, \dots, t_L$ are pairwise relatively prime with $\prod_{l=1}^{L} t_l \geq \frac{N}{s_1 u_1}$. Set

$$\begin{aligned}\delta &= \max_{\substack{\nu=0,\dots,u_m-1 \\ u_m \text{ hashes well}}} \left\{ \delta^{(m,\nu)} \right\} \\ &= \max_{\substack{\nu=0,\dots,u_m-1 \\ u_m \text{ hashes well}}} \left\{ \frac{1}{2dn} \left\| \mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu) \right\|_1 + \|\mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu)\|_1 \right\}.\end{aligned}$$

Then each $\omega \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ with $|c_\omega| > \varepsilon + 4\delta$ is added to the output R of Algorithm 1 in line 14, and its coefficient estimate from lines 10 and 11 satisfies

$$|x_R(\omega) - c_\omega| \leq 2\delta.$$

Proof. Let u_m be a good hashing prime and assume that $\omega \in (-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$ is contained in $R^{(m,\nu),\text{opt}}_{dn} \backslash R^{(m,\nu)}$, i.e., that it is one of the dn largest magnitude Fourier coefficient frequencies that are congruent to ν modulo u_m, but not contained in the set of frequencies returned by Algorithm 3 in [24] applied to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ for sparsity dn.

If $|c_\omega| \leq \varepsilon + 4\delta$, then $|c_\omega|$ is small enough that not including it in the reconstruction R does not yield large errors, since δ is defined as the maximum over the

$$\delta^{(m,\nu)} = \frac{1}{2dn} \left\| \mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu) \right\|_1 + \|\mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu)\|_1$$

for all good hashing primes u_m.

If $|c_\omega| > \varepsilon + 4 \cdot \delta \geq 4 \cdot \delta^{(m,\nu)}$ for all good hashing primes u_m, we know by Remark 3.8 (ii) that ω will be reconstructed more than $\frac{K}{2}$ times by Algorithm 3 in [24]. Then ω can only not be contained in $R^{(m,\nu)}$ if there are $dn+1$ frequencies $\tilde{\omega}$ in $R^{(m,\nu)} \backslash R^{(m,\nu),\text{opt}}_{dn}$ that satisfy $\left| x^{(m,\nu)}_{\tilde{\omega}} \right| \geq \left| x^{(m,\nu)}_{\omega} \right|$. Recall that u_m hashes all support sets $S_1, \dots, S_n$ well, so there are at most dn energetic frequencies congruent to ν modulo u_m. Suppose that all

frequencies with this residue are ordered by magnitude of their Fourier coefficient, i.e.,

$$\left|c_{\omega_1^{(m,\nu)}}\right| \geq \left|c_{\omega_2^{(m,\nu)}}\right| \geq \cdots \geq \left|c_{\omega_{dn}^{(m,\nu)}}\right| \geq \underbrace{\left|c_{\omega_{dn+1}^{(m,\nu)}}\right|}_{\leq \varepsilon} \geq \cdots \quad .$$

Then $|c_{\tilde{\omega}}| \leq \left|c_{\omega_{dn+1}^{(m,\nu)}}\right| \leq |c_\omega|$ for all $\tilde{\omega}$. By Remark 3.8 (iii) we have for all $\bar{\omega}$ that were reconstructed more than $\frac{K}{2}$ times by Algorithm 3 in [24] for ν modulo u_m that

$$\left|x_{\bar{\omega}}^{(m,\nu)} - c_{\bar{\omega}}\right| \leq \sqrt{2}\delta^{(m,\nu)}, \tag{9}$$

and further

$$\left|x_{\bar{\omega}}^{(m,\nu)}\right| \leq |c_{\bar{\omega}}| + \sqrt{2}\delta^{(m,\nu)} \tag{10}$$

$$\left|x_{\bar{\omega}}^{(m,\nu)}\right| \geq |c_{\bar{\omega}}| - \sqrt{2}\delta^{(m,\nu)}. \tag{11}$$

(9) - (11) hold for the ω and $\tilde{\omega}$ from above, and we find for all $\omega \in R_{dn}^{(m,\nu),\mathrm{opt}} \backslash R^{(m,\nu)}$ that

$$\begin{aligned}\left|c_{\omega_{dn+1}^{(m,\nu)}}\right| + \sqrt{2}\delta^{(m,\nu)} \geq |c_{\tilde{\omega}}| + \sqrt{2}\delta^{(m,\nu)} \geq \left|x_{\tilde{\omega}}^{(m,\nu)}\right| \\ \geq |x_\omega| \geq |c_\omega| - \sqrt{2}\delta^{(m,\nu)} \geq \left|c_{\omega_{dn+1}^{(m,\nu)}}\right| - \sqrt{2}\delta^{(m,\nu)}.\end{aligned}$$

Then

$$|c_\omega| \leq \left|c_{\omega_{dn+1}^{(m,\nu)}}\right| + 2\sqrt{2}\delta^{(m,\nu)} \leq \varepsilon + 2\sqrt{2}\delta^{(m,\nu)},$$

which contradicts $|c_\omega| > \varepsilon + 4\delta$, so we know that $\omega \in R^{(m,\nu)}$. Since this holds for all more than $\frac{M}{2}$ good hashing primes, ω will be considered from line 9 onward. We now prove the accuracy of the coefficient estimate. From (9) it follows that

$$\left|\mathrm{Re}\left(x_\omega^{(m,\nu)}\right) - \mathrm{Re}(c_\omega)\right| \leq \left|x_\omega^{(m,\nu)} - c_\omega\right| \leq \sqrt{2}\delta^{(m,\nu)} \leq \sqrt{2}\delta,$$

and analogously for the imaginary parts. As the estimates hold for more than $\frac{M}{2}$ of the hashing primes u_m, they also hold for the medians in lines 10 and 11. These are taken over the at most M coefficient estimates $x_{\tilde{\omega}}^{(m,\nu)}$ for ω, since for each u_m an $\tilde{\omega} = \omega$ can appear in at most one set $R^{(m,\nu)}$. Hence, we obtain

$$|\mathrm{Re}\,(x_\omega) - \mathrm{Re}(c_\omega)| \leq \sqrt{2}\delta \quad \text{and} \quad |\mathrm{Im}\,(x_\omega) - \mathrm{Im}(c_\omega)| \leq \sqrt{2}\delta,$$

and finally

$$|x_\omega - c_\omega| = \sqrt{(\mathrm{Re}\,(x_\omega - c_\omega))^2 + (\mathrm{Im}\,(x_\omega - c_\omega))^2} \leq \sqrt{\left(\sqrt{2}\delta\right)^2 + \left(\sqrt{2}\delta\right)^2} = 2\delta.$$

All that remains to be shown is that ω will actually be added to R in line 14. Similarly to (11) we find that $|x_\omega| \geq |c_\omega| - 2\delta$. Together with $|c_\omega| > \varepsilon + 4\delta$ this implies that $|x_\omega| > \varepsilon + 2\delta$. Then ω is only not included in the output if x_ω is not among the Bn

largest magnitude coefficient estimates, i.e., if there exist Bn other frequencies $\tilde{\omega}$ that satisfy $|x_{\tilde{\omega}}| \geq |x_\omega|$. We know that ω is energetic, which means that at least one of these $\tilde{\omega}$ must have a Fourier coefficient with $|c_{\tilde{\omega}}| \leq \varepsilon$. Then an analogue to (10) yields

$$|x_\omega| \leq |x_{\tilde{\omega}}| \leq |c_{\tilde{\omega}}| + 2\delta \leq \varepsilon + 2\delta,$$

which contradicts $|x_\omega| > \varepsilon + 2\delta$. Hence, ω will be added to R in line 14. □

Remark 3.11
One way to ensure that the requirements of Algorithm 1 are met is to define $t_1, \ldots, t_L$ as the L smallest primes satisfying

$$\prod_{l=1}^{L-1} t_l < \frac{N}{Bdn} \leq \prod_{l=1}^{L} t_l.$$

Set s_1 as the smallest prime that is greater than both dn and t_L,

$$s_1 := p_x > \max\{dn, t_L\} \geq p_{x-1}.$$

Instead of taking the minimal K, we can increase it slightly by using that $u_1 > B$, i.e.,

$$K = 8dn \left\lfloor \log_{s_1} \frac{N}{B} \right\rfloor + 1 \geq 8dn \left\lfloor \log_{s_1} \frac{N}{u_1} \right\rfloor + 1.$$

Hence, we can now choose the remaining s_k independently from the u_m to be $s_k := p_{x-1+k}$ for $k \in \{1, \ldots, K\}$. The hashing primes u_m can then be found by setting

$$u_1 := p_y > \max\{B, s_K\} \geq p_{y-1},$$

$M = 2(n+1)\lfloor \log_{u_1} N \rfloor + 1$ and $u_m := p_{y-1+m}$ for $m \in \{1, \ldots, M\}$. With these definitions $t_1, \ldots, t_L, s_1, \ldots, s_K, u_1, \ldots, u_M$ are pairwise relatively prime and satisfy that

$$s_k u_m \cdot \prod_{l=1}^{L} t_l \geq N,$$

as well as $s_k > dn$ and $u_m > B$ for all k and m. ◊

Using the s_k, t_l and u_m from Remark 3.11, the following main theorem gives us the runtime and error bounds of Algorithm 1.

Theorem 3.12
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be $P(n, d, B)$-structured sparse with noise η such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let $t_1, \ldots, t_L$ be the smallest primes with $\prod_{l=1}^{L} t_l \geq \frac{N}{Bdn}$. Set s_1 as the smallest prime greater than $\max\{dn, t_L\}$, $K = 8dn\lfloor \log_{s_1} \frac{N}{B} \rfloor + 1$ and $s_2, \ldots, s_K$ as the first $K - 1$ primes greater than s_1. Let u_1 be the smallest prime greater than $\max\{B, s_K\}$, $M = 2(n+1)\lfloor \log_{u_1} N \rfloor + 1$ and $u_2, \ldots, u_M$ the first $M - 1$ primes greater than u_1. Let further δ be defined as

$$\delta := \max_{\substack{\nu=0,\ldots,u_m-1 \\ u_m \text{ hashes well}}} \left\{ \frac{1}{2dn} \|\mathbf{c}(N, u_m, \nu) - \mathbf{c}^{\text{opt}}_{2dn}(N, u_m, \nu)\|_1 + \|\mathbf{c}(N, \mathbb{Z}, u_m, \nu) - \mathbf{c}(u_m, \nu)\|_1 \right\}.$$

Then the output $(R, \mathbf{x}_R)$ of Algorithm 1 satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2 + \sqrt{Bn} \cdot (\varepsilon + 6\delta).$$

If $B > s_K$, the output can be computed in a runtime of

$$\mathcal{O}\left(\frac{d^2 n^3 (B + n\log N) \cdot \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log N \log\left(dn \log \frac{N}{B}\right) \log^2\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B \log^2(dn) \log\log \frac{N}{Bdn}}\right)$$

and the algorithm has a sampling complexity of

$$\mathcal{O}\left(\frac{d^2 n^3 (B + n\log N) \cdot \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log N \log\left(dn \log \frac{N}{B}\right) \log\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B \log^2(dn) \log\log \frac{N}{Bdn}}\right).$$

Proof. For the vector $\mathbf{c}_R(N)$, whose entries are the Fourier coefficients c_ω for the frequencies contained in R and zero otherwise, it always holds that

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2 + \|\mathbf{c}_R(N) - \mathbf{x}_R\|_2. \tag{12}$$

The square of first summand in (12) can be written as

$$\begin{aligned}
&\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 = \sum_{\omega=-\lceil \frac{N}{2} \rceil + 1}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - c_R(\omega)|^2 \\
&= \sum_{\omega \notin R} |c_\omega|^2 + \sum_{\omega \in R} \underbrace{|c_\omega - c_R(\omega)|^2}_{=0} = \sum_{\omega \notin R_{Bn}^{\mathrm{opt}}} |c_\omega|^2 + \sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2 - \sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2 \\
&= \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2^2 + \sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2 - \sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2.
\end{aligned}$$

For every $\omega \in R_{Bn}^{\mathrm{opt}} \setminus R$ we know by Lemma 3.10 that $|c_\omega| \leq \varepsilon + 4\delta$, because otherwise it would be contained in R. Since $R_{Bn}^{\mathrm{opt}} \setminus R$ contains at most Bn elements, this yields

$$\begin{aligned}
\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 &= \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2^2 + \underbrace{\sum_{\omega \in R_{Bn}^{\mathrm{opt}} \setminus R} |c_\omega|^2}_{\leq Bn(\varepsilon+4\delta)^2} - \underbrace{\sum_{\omega \in R \setminus R_{Bn}^{\mathrm{opt}}} |c_\omega|^2}_{\geq 0} \\
&\leq \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2^2 + Bn(\varepsilon + 4\delta)^2.
\end{aligned}$$

For the second summand in (12) consider an $\omega \in R$. For each of the more than $\frac{M}{2}$ good hashing primes it has to be contained in exactly one of the $R^{(m,\nu)}$, so ω must have been reconstructed more than $\frac{K}{2}$ times by Algorithm 3 in [24], applied to the entries congruent to $\nu \equiv \omega \mod u_m$. Hence we have by (9) that

$$\left|x_\omega^{(m,\nu)} - c_\omega\right| \leq \sqrt{2}\delta^{(m,\nu)} \quad \text{and} \quad |x_R(\omega) - c_\omega| \leq 2\delta,$$

analogously to the proof of Lemma 3.10. Since R contains at most Bn elements, we find

$$\|\mathbf{c}_R(N) - \mathbf{x}_R\|_2^2 = \sum_{\omega \in R} \underbrace{|c_\omega - x_R(\omega)|^2}_{\leq 4\delta^2} \leq 4Bn\delta^2.$$

Combining all these estimates we obtain that

$$\begin{aligned}\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq& \sqrt{\|\mathbf{c}(N) - \mathbf{c}_R(N)\|_2^2 + \|\mathbf{c}_R(N) - \mathbf{x}_R\|_2^2} \\ \leq& \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2 + \sqrt{Bn}(\varepsilon + 4\delta) + 2\sqrt{Bn}\delta \\ =& \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_2 + \sqrt{Bn}(\varepsilon + 6\delta).\end{aligned}$$

In order to determine the runtime of the algorithm let us first consider the runtime of the calculation of the DFTs in line 2. It was shown in [24, 27] that

$$t_L = \mathcal{O}\left(\log \frac{N}{Bdn}\right) \quad \text{and} \quad s_K = \mathcal{O}\left(dn \log_{dn} \frac{N}{B} \log\left(dn \log \frac{N}{B}\right)\right).$$

Let π be the prime-counting function,

$$\pi(x) = \sum_{\substack{2 \leq p \leq x \\ p \text{ prime}}} 1.$$

If $s_K \leq B$, we can set $u_1 := p_y > B \geq p_{y-1}$ to be the first prime greater than B, so by the Prime Number Theorem (see [36])

$$y - 1 = \pi(B) = \mathcal{O}\left(\frac{B}{\log B}\right)$$

and

$$y - 1 + M = \mathcal{O}\left(\frac{B}{\log B} + n \log_B N\right) = \mathcal{O}\left(\frac{B + n \log N}{\log B}\right).$$

An equivalent formulation of the Prime Number Theorem yields for $u_M = p_{y-1+M}$ that

$$u_M = \mathcal{O}((y - 1 + M)\log(y - 1 + M)) = \mathcal{O}\left(\frac{B + n \log N}{\log B} \log\left(\frac{B + n \log N}{\log B}\right)\right).$$

In [27] it was proven that

$$\sum_{\substack{2 \leq p \leq R \\ p \text{ prime}}} p = \mathcal{O}\left(\frac{R^2}{\log R}\right).$$

Because estimating $\sum_{m=1}^{M} u_m \log u_m$ by summing $p \log p$ for all primes less than u_M would take into account many primes that do not contribute to the sum, we use instead that

$$\sum_{m=1}^{M} u_m \log u_m = \mathcal{O}(M \cdot u_M \log u_M).$$

In line 2 we have to calculate the DFTs of length $s_k t_l u_m$ of the vectors $\mathbf{A}_{s_k t_l u_m}$ for all k, l, m. Even if some $\tilde{m} \in \mathbb{N}$ is not a power of 2, computing a DFT of length $\tilde{m}$ has a

runtime of $\mathcal{O}(\tilde{m}\log\tilde{m})$ (see [5, 40]). Since $u_1 > s_K$, we obtain a computational effort of

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L}\sum_{m=1}^{M} s_k t_l u_m \log(s_k t_l u_m)\right) = \mathcal{O}\left(\sum_{k=1}^{K} s_k \sum_{l=0}^{L} t_l \sum_{m=1}^{M} u_m \log u_m\right)\\
=&\mathcal{O}\left(\frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\cdot M\cdot u_M\log u_M\right)\\
=&\mathcal{O}\left(\frac{d^2n^3(B+n\log N)\cdot\log^2\frac{N}{Bdn}\log^2\frac{N}{B}\log N\log(dn\log\frac{N}{B})\log^2\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B\log^2(dn)\log\log\frac{N}{Bdn}}\right).
\end{aligned}$$

For the sampling complexity we find

$$\begin{aligned}
&\mathcal{O}\left(\sum_{k=1}^{K} s_k \sum_{l=0}^{L} t_l \sum_{m=1}^{M} u_m\right) = \mathcal{O}\left(\frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\cdot M\cdot u_M\right)\\
=&\mathcal{O}\left(\frac{d^2n^3(B+n\log N)\cdot\log^2\frac{N}{Bdn}\log^2\frac{N}{B}\log N\log(dn\log\frac{N}{B})\log\left(\frac{B+n\log N}{\log B}\right)}{\log^2 B\log^2(dn)\log\log\frac{N}{Bdn}}\right).
\end{aligned}$$

Now we can estimate the runtime of the remaining steps of the algorithm. The q in line 1 does not actually have to be computed, it is just defined there in order to introduce more readable notation. In line 6 we apply Algorithm 3 in [24] to the column corresponding to the residue ν modulo u_m of $\mathcal{G}_{L,K}\cdot(\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T$, which was already computed in line 2. We know from Remark 3.8 that the runtime of Algorithm 3 in [24] is dominated by the computation of the DFTs, so the runtime of lines 4 to 8 of Algorithm 1 is dominated by the runtime of line 2. In order to find out for which frequencies line 9 to 12 have to be executed, we can sort the $2dn\sum_{m=1}^{M} u_m$ frequencies that are returned by all the calls of Algorithm 3 in [24] by size and count how often each distinct frequency appears. This can be done in

$$\mathcal{O}\left(2dn\left(\sum_{m=1}^{M} u_m\right)\cdot\log\left(2dn\sum_{m=1}^{M} u_m\right)\right) = \mathcal{O}\left(dnMu_M\cdot\log(dnMu_M)\right)$$

time, so it is also insignificant compared to the DFT computation. There are at most

$$\frac{2}{M}\cdot\sum_{m=1}^{M}\sum_{\nu=0}^{u_m-1} 2dn = \frac{4dn}{M}\cdot\sum_{m=1}^{M} u_m = \mathcal{O}\left(4dn\cdot u_M\right)$$

frequencies that can have been found more than $\frac{M}{2}$ times. If we fix one of these frequencies, ω, then for each u_m there is exactly one residue $\nu^{(m)} \in \{0,\dots,u_m-1\}$ with $\omega \equiv \nu^{(m)} \bmod u_m$. Since the $2dn$ frequencies recovered for any fixed residue modulo some hashing prime are distinct, there can be at most M frequencies $\tilde{\omega}$ satisfying $\tilde{\omega} = \omega$ found for all hashing primes. This means that the medians in lines 10 and 11 are taken over at most M elements. As medians can be computed by sorting, both lines have a runtime of $\mathcal{O}(M\log M)$. Combining these considerations we obtain that lines 9 to 12 require

$$\mathcal{O}\left(4dn\cdot u_M\cdot M\log M\right)$$

arithmetical operations, which is dominated by the effort of the DFT computations in

line 2. Finally, sorting the $\mathcal{O}(4dn \cdot u_M)$ coefficient estimates in line 13 has a runtime of

$$\mathcal{O}(4dnu_M \log(4dnu_M)),$$

so, as stated above, the runtime of Algorithm 1 is determined by the one of line 2. □

If $f + \eta$ is bandlimited, simplifying the above error bound yields Theorem 1.2 in §1.4.

Corollary 3.13
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be $P(n, d, B)$-structured sparse with noise η such that $\mathbf{c}(\eta) \in \ell^1$, $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ and f and $f + \eta$ are bandlimited to $(-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$. Choosing the s_k, t_l, u_m as in Theorem 3.12, the output $(R, \mathbf{x}_R)$ of Algorithm 1 satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_2 \leq \left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_2 + \sqrt{Bn}\left(\varepsilon + \frac{3}{dn}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{2Bn}(N)\right\|_1\right).$$

Proof. By definition of δ we have that

$$\begin{aligned}\delta :=& \max_{\substack{\nu=0,\dots,u_m-1 \\ u_m \text{ hashes well}}} \left\{\delta^{(m,\nu)}\right\} = \delta^{(m',\nu')} \\ =& \frac{1}{2dn}\left\|\mathbf{c}(N, u_{m'}, \nu') - \mathbf{c}^{\text{opt}}_{2dn}(N, u_{m'}, \nu')\right\|_1 + \left\|\mathbf{c}(N, \mathbb{Z}, u_{m'}, \nu') - \mathbf{c}(u_{m'}, \nu')\right\|_1\end{aligned}$$

for some residue ν' modulo a good hashing prime $u_{m'}$. Since f and $f+\eta$ are bandlimited, the second summand is 0. Due to the fact that f is $P(n, d, B)$-structured sparse, any restriction $\mathbf{c}(N, u_{m'}, \nu)$ of $\mathbf{c}(f)$ to the frequencies that are congruent to ν modulo $u_{m'}$ is dn-sparse. As there are at most Bn energetic frequencies, we find the following estimate,

$$\begin{aligned}\delta \leq& \sum_{\nu=0}^{u_{m'}-1} \delta^{(m',\nu)} \leq \sum_{\nu=0}^{u_{m'}-1} \frac{1}{2dn}\left\|\mathbf{c}(N, u_{m'}, \nu) - \mathbf{c}^{\text{opt}}_{2dn}(N, u_{m'}, \nu)\right\|_1 \\ \leq& \frac{1}{2dn}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{2Bn}(N)\right\|_1,\end{aligned}$$

so the error bound from Theorem 3.12 reduces to

$$\begin{aligned}\|\mathbf{c}(N) - \mathbf{x}_R\|_2 &\leq \left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_2 + \sqrt{Bn} \cdot (\varepsilon + 6\delta) \\ &\leq \left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{Bn}(N)\right\|_2 + \sqrt{Bn} \cdot \left(\varepsilon + \frac{3}{dn}\left\|\mathbf{c}(N) - \mathbf{c}^{\text{opt}}_{2Bn}(N)\right\|_1\right).\end{aligned}$$

□

4 Algorithm for Functions with Simplified Fourier Structure

The algorithm introduced in §3.3 always uses M hashing primes of which more than $M/2$ are good. If we can guarantee that one hashing prime suffices, a simplified, faster version of Algorithm 1 can be applied, which is what we will study in the following.

4.1 Structured Sparse Functions Requiring Only One Hashing Prime

If certain additional information about the polynomials generating the support sets $S_1, \dots, S_n$ is known, the number of required hashing primes can be reduced to one. We know by Lemma 2.3 that a prime u does not hash a support set S_j well if and only if u divides all non-constant coefficients. Thus we can make the following observation.

Theorem 4.1
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be $P(n, d, B)$-structured sparse with noise η such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Let the support set $S = \bigcup_{j=1}^{n} S_j$ of f be defined by the non-constant polynomials $P_j(x) = \sum_{k=0}^{d} a_{jk} x^k$ for $j \in \{1, \dots, n\}$. Let $u > B$ be a prime such that for all $j \in \{1, \dots, n\}$ there exists a $k_j \in \{1, \dots, d\}$ with $p \nmid a_{jk_j}$. Then u hashes all support sets well. Set $M = 1$ and the s_k and t_l as in Theorem 3.12. If $B > s_K$, the runtime of Algorithm 1 reduces to

$$\mathcal{O}\left(\frac{u \log u \cdot (dn)^2 \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log\left(dn \log \frac{N}{B}\right)}{\log^2(dn) \log\log \frac{N}{Bdn}}\right),$$

while only

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log\left(dn \log \frac{N}{B}\right)}{\log^2(dn) \log\log \frac{N}{Bdn}}\right)$$

samples of $f + \eta$ are being used. If $B \leq s_K$, we obtain a runtime of

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log^2\left(dn \log \frac{N}{B}\right)}{\log^2(dn) \log\log \frac{N}{Bdn}}\right)$$

and a sampling complexity of

$$\mathcal{O}\left(\frac{u \cdot (dn)^2 \cdot \log^2 \frac{N}{Bdn} \log^2 \frac{N}{B} \log\left(dn \log \frac{N}{B}\right)}{\log^2(dn) \log\log \frac{N}{Bdn}}\right).$$

Proof. Lemma 2.3 implies that u hashes all n support sets well, so the restriction to the frequencies congruent to ν modulo u is at most dn-sparse for all residues. Hence, we can apply Algorithm 3 in [24] to $\mathcal{G}_{L,K} \cdot (\mathcal{H}_M \circledast (\widehat{\mathbf{A}_q})^T)^T_{\boldsymbol{\rho}^T_{m,\nu}}$ for every residue ν modulo u, and will always obtain a good reconstruction. As there are no residues modulo which more than dn energetic frequencies can collide, it suffices to set $u_1 = u$ and $M = 1$. Lines 9 to 12 do not have to be executed, since every frequency will be recovered in line 6 for exactly one residue.

If $u > s_K$, we can define the t_l and s_k as in Remark 3.11. If $u \leq s_K$, then u might collide with one of the s_k or t_l. In that case we shift all the t_l and s_k, starting by u, to the next largest prime, so they are at most the next largest prime greater than the original t_l and s_k. This does not change the estimates in the proof of Theorem 3.12.

Let us first consider the case that $u > s_K$. We obtain for the computation of the DFTs

in line 2, which dominates the runtime of Algorithm 1, that they require

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log(s_k t_l u)\right) = \mathcal{O}\left(u\log u \cdot \frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\right)\\ =&\mathcal{O}\left(\frac{u\log u\cdot (dn)^2 \log^2 \frac{N}{Bdn}\log^2\frac{N}{B}\log\left(dn\log\frac{N}{B}\right)}{\log^2(dn)\log\log\frac{N}{Bdn}}\right)\end{aligned}$$

arithmetical operations and have a sampling complexity of

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) = \mathcal{O}\left(u\cdot \frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\right)\\ =&\mathcal{O}\left(\frac{u\cdot (dn)^2 \log^2 \frac{N}{Bdn}\log^2\frac{N}{B}\log\left(dn\log\frac{N}{B}\right)}{\log^2(dn)\log\log\frac{N}{Bdn}}\right).\end{aligned}$$

If $u \leq s_K$, we obtain a runtime of

$$\begin{aligned}&\mathcal{O}\left(u\cdot\sum_{l=0}^{L} t_l \sum_{k=1}^{K} s_k \log s_k\right) = \mathcal{O}\left(u\cdot \frac{t_L^2}{\log t_L}\cdot s_K^2\right)\\ =&\mathcal{O}\left(\frac{u\cdot (dn)^2\cdot \log^2 \frac{N}{Bdn}\log^2\frac{N}{B}\log^2\left(dn\log\frac{N}{B}\right)}{\log^2(dn)\log\log\frac{N}{Bdn}}\right)\end{aligned}$$

and a sampling complexity of

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) = \mathcal{O}\left(u\cdot \frac{t_L^2}{\log t_L}\cdot\frac{s_K^2}{\log s_K}\right)\\ =&\mathcal{O}\left(\frac{u\cdot (dn)^2\cdot \log^2 \frac{N}{Bdn}\log^2\frac{N}{B}\log\left(dn\log\frac{N}{B}\right)}{\log^2(dn)\log\log\frac{N}{Bdn}}\right).\end{aligned}$$

□

We now give some conditions on the coefficients of the polynomials $P_1, \ldots, P_n$ generating the support sets $S_1, \ldots, S_n$ which guarantee that all S_j are hashed well. All of the conditions arise by tightening the necessary and sufficient requirement of the existence of a non-constant coefficient that is not divisible by u in Theorem 4.1. Hence, all of the conditions are sufficient, but they may not be necessary anymore, which might make them easier to prove in practice.

Lemma 4.2
Let f be $P(n, d, B)$-structured sparse. In the following cases any prime $u > B$ is guaranteed to hash all frequency subsets well,

(i) $\forall\, j \in \{1, \ldots, n\}\colon \gcd(a_{j1}, \ldots, a_{jd}) < B$, which includes $\gcd(a_{j_1}, \ldots, a_{jd}) = 1$,

(ii) $\forall\, j \in \{1, \ldots, n\}\, \exists\, k_j \in \{1, \ldots, d\}\colon \left|a_{jk_j}\right| < B$,

(iii) $\forall\, j \in \{1, \ldots, n\}\, \exists\, k_j \in \{1, \ldots, d\}\colon a_{jk_j} = 1$, which includes monic polynomials,

(iv) $\forall\, j \in \{1, \ldots, n\}\colon \deg(P_j) = 1$ and $a_{j1} = 1$, which is the block sparse case.

If we have already fixed a prime $u > B$ that is supposed to be the hashing prime, the following conditions imply that u indeed hashes all support sets well.

Lemma 4.3
Let f be $P(n, d, B)$-structured sparse. In the following cases a fixed prime $u > B$ is guaranteed to hash all support sets well.

(v) (i) to (iv) from Lemma 4.2 hold for u, and B can be changed to u for (i) and (ii)

(vi) $\forall\, j \in \{1, \dots, n\}\colon u \nmid \sum_{k=1}^{d} a_{jk}$,

(vii) $\forall\, j \in \{1, \dots, n\}\, \exists\, \varepsilon_j \in \{0,1\}^d\colon u \nmid \sum_{k=1}^{d} (-1)^{\varepsilon_{jk}} a_{jk}$.

4.2 Block Frequency Sparse Functions

Let us consider block frequency sparse functions (condition (iv) in Lemma 4.2) in more detail. In that case, the support sets $S_1, \dots, S_n$ are of the form

$$S_j = \{a_{j0}, a_{j0} + 1, \dots, a_{j0} + B - 1\}\,, \quad j \in \{1, \dots, n\},$$

and we can improve the runtime of our algorithm even further.

Definition 4.4 ((n, B)-block Sparsity)
A $P(n, 1, B)$-structured sparse function f is called (n, B)-*block sparse* if the support sets $S_1, \dots, S_n$ are generated by the monic linear polynomials

$$P_j(x) := x + a_j, \quad j \in \{1, \dots, n\}.$$

For block sparse functions we can extend the definition of good hashing primes to integers, because we do not require the multiplicative invertibility of all nonzero elements anymore.

Definition 4.5
Let f be (n, B)-block sparse with support set $S = \bigcup_{j=1}^{n} S_j$ generated by the polynomials $P_1, \dots, P_n$. An integer $u > B$ *hashes* a support set S_j *well* if

$$|\{\omega \bmod u : \omega \in S_j\}| = B \quad \forall j \in \{1, \dots, n\}.$$

Remark 4.6
For an (n, B)-block sparse function f any integer $u > B$ hashes every support set S_j well, since it consists of B consecutive frequencies. Thus for every residue ν modulo u the restriction of S to the frequencies congruent to ν is at most n-sparse,

$$|\{\omega \equiv \nu \bmod u\colon \omega \in S\}| \leq n \quad \forall \nu \in \{0, \dots, u-1\}.$$

$\Diamond$

If f is (n, B)-block sparse, we can choose the hashing integer u to be the smallest power of 2 that is greater than the block length B. Then $u = \mathcal{O}(B)$, which allows us to give better runtime estimates. Additionally, computing DFTs of length $s \cdot t \cdot u$, where s and t are small primes and u is a power of 2 is faster than if u were a prime of the same size.

Corollary 4.7
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be (n, B)-block sparse with noise η such that $\mathbf{c}(\eta) \in \ell^1$ and $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$. Set $u := 2^\alpha$, where $\alpha := \lfloor \log_2 B \rfloor + 1$, $M = 1$ and the s_k and t_l as in Theorem 3.12. If $u > s_K$, the runtime of Algorithm 1 is given by

$$\mathcal{O}\left(\frac{B \log B \cdot n^2 \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right),$$

and otherwise, if $u < s_K$, by

$$\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log^2\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right).$$

In both cases the algorithm has a sampling complexity of

$$\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right).$$

Proof. Choosing u as a power of 2 implies that we now have to slightly modify the t_l and s_k. Similar to the choice of the primes in Remark 3.11, we take the smallest L odd primes such that their product is greater than or equal to $\frac{N}{un}$,

$$\prod_{l=1}^{L-1} t_l < \frac{N}{un} \leq \prod_{l=1}^{L} t_l, \quad t_1 := 3.$$

This means that $t_l = p_{l+1}$. Let s_1 be the smallest prime that is greater than n and t_L,

$$s_1 := p_x > \max\{n, t_L\} \geq p_{x-1}.$$

In this setting we can use the minimal K,

$$K = 8n \left\lfloor \log_{s_1} \frac{N}{u} \right\rfloor + 1.$$

The remaining s_k can be set as $s_k := p_{x-1+k}$ for $k \in \{1, \ldots, K\}$. Then the set $\{t_1, \ldots, t_L, s_1, \ldots, s_K, u\}$ is pairwise relatively prime, $u > B$ and

$$\prod_{l=1}^{L} t_l \geq \frac{N}{s_1 u_1},$$

so the CRT can be applied. Since we chose $t_1 = 3$, the prime t_L in this case is at most the smallest prime greater than the t_L from Remark 3.11 for $d = 1$, and thus we still have that

$$t_L = \mathcal{O}\left(\log \frac{N}{un}\right) \quad \text{and} \quad s_K = \mathcal{O}\left(n \log_n \frac{N}{u} \log\left(n \log \frac{N}{u}\right)\right).$$

Using additionally that $u = \mathcal{O}(B)$, the runtime of Algorithm 1 for $u > s_K$ is given by

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log(s_k t_l u)\right) = \mathcal{O}\left(u \log u \cdot \frac{s_K^2}{\log s_K} \cdot \frac{t_L^2}{\log t_L}\right)\\ =&\mathcal{O}\left(\frac{B \log B \cdot n^2 \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right).\end{aligned}$$

If $u < s_K$, we obtain a runtime of

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u \log(s_k t_l u)\right) = \mathcal{O}\left(u \cdot s_K^2 \cdot \frac{t_L^2}{\log t_L}\right)\\ =&\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log^2\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right).\end{aligned}$$

In both cases the number of required samples of $f + \eta$ is

$$\begin{aligned}&\mathcal{O}\left(\sum_{k=1}^{K}\sum_{l=0}^{L} s_k t_l u\right) = \mathcal{O}\left(u \cdot \frac{s_K^2}{\log s_K} \cdot \frac{t_L^2}{\log t_L}\right)\\ =&\mathcal{O}\left(\frac{Bn^2 \cdot \log^2 \frac{N}{Bn} \log^2 \frac{N}{B} \log\left(n \log \frac{N}{B}\right)}{\log^2 n \log\log \frac{N}{Bn}}\right).\end{aligned}$$

□

If $f + n$ is bandlimited, we can prove the 1-norm error bound in Theorem 1.2 in §1.4.

Corollary 4.8
Let $N \in \mathbb{N}$ and $f\colon [0, 2\pi] \to \mathbb{C}$ be (n, B)-block sparse with noise η such that $\eta \in \ell^1$, $\|\mathbf{c}(\eta)\|_\infty \leq \varepsilon$ and f and $f + \eta$ are bandlimited to $(-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$. Setting $u := u_1 := 2^\alpha$, where $\alpha := \lfloor \log_2 B \rfloor + 1$, $M = 1$ and the s_k and t_l as in Theorem 3.12, the output $(R, \mathbf{x}_R)$ of Algorithm 1 satisfies

$$\|\mathbf{c}(N) - \mathbf{x}_R\|_1 \leq 4 \left\|\mathbf{c}(N) - \mathbf{c}_{Bn}^{\mathrm{opt}}(N)\right\|_1 + 2Bn\varepsilon.$$

Proof. As we do not have to take medians over the estimates obtained for the different hashing primes, we can consider the following inequality,

$$\begin{aligned}&\|\mathbf{c}(N) - \mathbf{x}_R\|_1 = \sum_{\omega=-\lceil \frac{N}{2} \rceil + 1}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - x_\omega| = \sum_{\nu=0}^{u-1} \sum_{\substack{\omega=-\lceil \frac{N}{2} \rceil + 1 \\ \omega \equiv \nu \bmod u}}^{\lfloor \frac{N}{2} \rfloor} |c_\omega - x_\omega|\\ =&\sum_{\nu=0}^{u-1}\Bigg(\sum_{\substack{\omega \in R^{(1,\nu)} \\ \omega \equiv \nu \bmod u}} |c_\omega - x_\omega| + \sum_{\substack{\omega \notin R^{(1,\nu)} \\ \omega \equiv \nu \bmod u}} |c_\omega|\Bigg)\\ =&\sum_{\nu=0}^{u-1}\Bigg(\sum_{\omega \in R^{(1,\nu)}} |c_\omega - x_\omega| + \sum_{\omega \notin R_n^{(1,\nu),\mathrm{opt}}} |c_\omega| + \sum_{\omega \in R_n^{(1,\nu),\mathrm{opt}} \setminus R^{(1,\nu)}} |c_\omega| - \sum_{\omega \in R^{(1,\nu)} \setminus R_n^{(1,\nu),\mathrm{opt}}} |c_\omega|\Bigg).\end{aligned}$$

By (9) the $2n$ elements of $R^{(1,\nu)}$ satisfy $|c_\omega - x_\omega| \leq \sqrt{2}\delta^{(1,\nu)}$. From the proof of Lemma 3.10 it follows that $|c_\omega| \leq \varepsilon + 2\sqrt{2}\delta^{(1,\nu)}$ for all $\omega \in R_n^{(1,\nu),\text{opt}} \backslash R^{(1,\nu)}$, because otherwise ω would be added to $R^{(1,\nu)}$. Recall the definition of $\delta^{(1,\nu)}$,

$$\delta^{(1,\nu)} := \frac{1}{2n} \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_{2n}^{\text{opt}}(N,u,\nu) \right\|_1 + \underbrace{\|\mathbf{c}(N,\mathbb{Z},u,\nu) - \mathbf{c}(u,\nu)\|_1}_{=0},$$

since $f+\eta$ is bandlimited. Combining these considerations we find that

$$\begin{aligned}
\|\mathbf{c}(N) - \mathbf{x}_R\|_1 &\leq \sum_{\nu=0}^{u-1} \left(2\sqrt{2}n\delta^{(1,\nu)} + \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_n^{\text{opt}}(N,u,\nu) \right\|_1 + n\left(\varepsilon + 2\sqrt{2}\delta^{(1,\nu)}\right)\right) \\
&= \sum_{\nu=0}^{u-1} \left(\left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_n^{\text{opt}}(N,u,\nu) \right\|_1 + 4\sqrt{2}n \left(\frac{1}{2n} \left\| \mathbf{c}(N,u,\nu) - \mathbf{c}_{2n}^{\text{opt}}(N,u,\nu) \right\|_1 \right)\right) \\
&\quad + nu\varepsilon.
\end{aligned}$$

Because f is (n,B)-block sparse, every restriction $\mathbf{c}(N,u,\nu)$ of $\mathbf{c}(f+\eta)$ to the frequencies that are congruent to ν modulo u is n-sparse, so we can use the same idea as in the proof of Corollary 3.13 to obtain

$$\begin{aligned}
\|\mathbf{c}(N) - \mathbf{x}_R\|_1 &\leq \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\text{opt}}(N) \right\|_1 + 2\sqrt{2} \cdot \left\| \mathbf{c}(N) - \mathbf{c}_{2Bn}^{\text{opt}}(N) \right\|_1 + nu\varepsilon \\
&\leq 4 \cdot \left\| \mathbf{c}(N) - \mathbf{c}_{Bn}^{\text{opt}}(N) \right\|_1 + 2Bn\varepsilon.
\end{aligned}$$

□

5 Numerical Evaluation

In this section we evaluate the performance of two different variants of Algorithm 1 including (*i*) the deterministic variant for block sparse functions described in §4.2 (referred to as the **F**ourier **A**lgorithm for **S**tructured sparsi**T**y (FAST) below), and (*ii*) a randomized implementation of Algorithm 1 which only utilizes a small random subset of the M hashing primes used by FAST for each choice of its parameters (referred to as the **F**ourier **A**lgorithm for **S**tructured sparsi**T**y with **R**andomization (FASTR) below). Both of these C++ implementations are publicly available.[3] We also compare these implementations' runtime and robustness characteristics with GFFT,[4] FFTW 3.3.4,[5] and sFFT 2.0.[6]

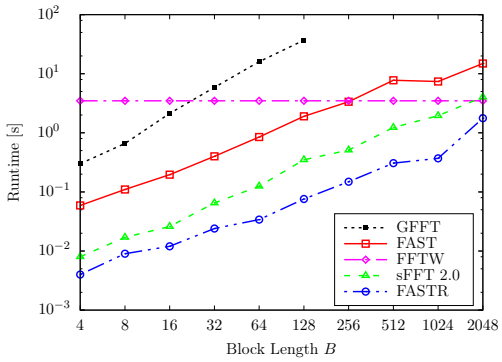
Note that FAST and FASTR are both designed to approximate functions that are $(n,B)-$block sparse in Fourier space. This means that both FAST and FASTR take upper bounds on the number of blocks, n, and length of each block, B, present in the spectrum of the functions they aim to recover as parameters. In contrast, both GFFT (a deterministic sparse Fourier transform [41]) and SFFT 2.0 (a randomized noise robust sparse Fourier transform [19]) only require an upper bound on the effective sparsity, s, of the function's Fourier coefficients. Herein s is always set so that $s = Bn$ for

[3] `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`.

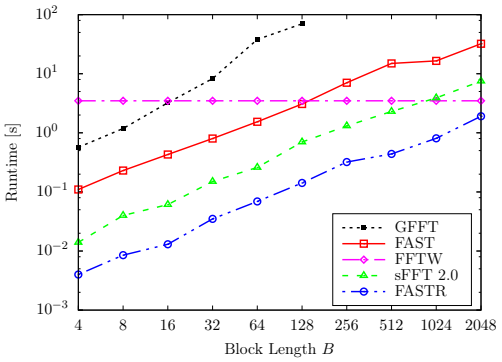
[4] Also available at `http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software`.

[5] `http://www.fftw.org/`

[6] `https://groups.csail.mit.edu/netmit/sFFT/`



(a) Runtime comparison for bandwidth $N = 2^{26}$ and $n = 2$ blocks.



(b) Runtime comparison for bandwidth $N = 2^{26}$ and $n = 3$ blocks.

Figure 1: Runtime plots for several algorithms and implementations of sparse Fourier transform for different B settings

these methods. Finally, FFTW is a highly optimized and publicly available implementation of the traditional FFT algorithm which runs in $\mathcal{O}(N \log N)$-time for input vectors of length N. All the FFTW results below were obtained using FFTW 3.3.4 with its FFTW_MEASURE plan.

For the runtime experiments below the trial signals were formed by choosing sets of frequencies with $(n, B)-$block sparsity uniformly at random from $(-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$. Each frequency in this set was then assigned a magnitude 1 Fourier coefficient with a uniformly random phase. The remaining frequencies were all set to zero. Every data point in a figure below corresponds to an average over 100 trial runs on 100 different trial signals of this kind. For different n, B and N, the parameters in each randomized algorithm (i.e. FASTR and sFFT 2.0) were chosen so that the probability of correctly recovering an (n, B)-block sparse function was at least 0.9 for each run. Finally, all experiments were run on a Linux CentOS machine with 2.50GHz CPU and 16 GB of RAM.

5.1 Runtime as Block Length B Varies: $N = 2^{26}$, $n = 2$ and $n = 3$

In Figure 1a we fix the number of blocks to $n = 2$ and the bandwidth to $N = 2^{26}$, and then perform numerical experiments for 10 different block lengths $B = 2^2$, 2^3, ..., 2^{11}. We then plot the runtime (averaged over 100 trial runs) for FAST, FASTR, GFFT, sFFT 2.0 and FFTW. As expected, the runtime of FFTW is constant with increasing sparsity. The runtimes of all the sparse Fourier transform algorithms other than GFFT are approximately linear in B, and they have similar slopes. Figure 1a demonstrates that allowing a small probability of incorrect recovery always lets the randomized algorithms (FASTR and sFFT 2.0) outperform the deterministic algorithms with respect to runtime. Among the deterministic algorithms, FAST is always faster than GFFT, and only becomes slower than FFTW when the value of B is greater than 256. The runtimes of both FASTR and sFFT 2.0 are still comparable with the one of FFTW when the block length B is 2048. Comparing with sFFT 2.0, FASTR has better runtime performance on these block sparse functions, and is the only algorithm that is still faster than FFTW when $B = 2048$. In Figure 1b we use the same settings of N and B as in the previous experiment and increase the number of blocks n from 2 to 3. With these settings the

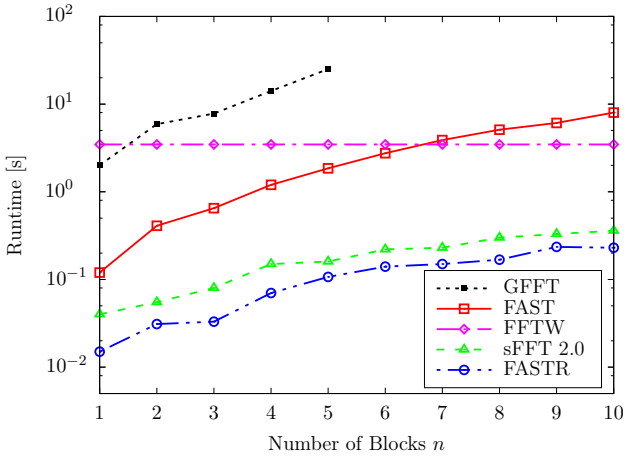


Figure 2: Runtime comparison for bandwidth $N = 2^{26}$ and block length $B = 32$.

largest sparsity $s = Bn$ increases from 4048 $(2 \cdot 2^{11})$ to 6144 $(3 \cdot 2^{11})$. The respective results for the methods are similar in this plot.

5.2 Runtime as Number of Blocks n Varies: $N = 2^{26}$ and $B = 32$

In Figure 2 we fix the bandwidth $N = 2^{26}$ and block length $B = 32$, then vary the number of blocks n from 1 to 10. Looking at Figure 2, we can see that the deterministic sparse FFTs, GFFT and FAST, both have runtimes that increase more rapidly with n than those of their randomized competitors. Among the three deterministic algorithms, FAST has the best performance when the number of blocks is smaller than 6. Similar to the previous experiments, FFTW becomes the fastest deterministic algorithm when the sparsity $s = Bn$ gets large enough (greater than 224 in this experiment). The two randomized algorithms are both faster than FFTW by an order of magnitude when the number of blocks is 10. Similarly, FASTR is always faster than sFFT 2.0 for the examined value of N.

5.3 Runtime as Signal Size N Varies: $n = 2$ and $B = 64$

In Figure 3 we fix the number of blocks $n = 2$ and block length $B = 64$, then test the performance of the different algorithms with various bandwidths N. It can be seen in Figure 3 that FFTW is the fastest deterministic algorithm for small bandwidth values. However, the runtime of FFTW becomes slower than the one of FAST when the bandwidth N is greater than 2^{24}. GFFT is the slowest deterministic algorithm for this sparsity level for all plotted N. Comparing randomized SFT algorithms, FASTR always performs better than sFFT 2.0 when the bandwidth is greater than 2^{18}.

5.4 Robustness to Noise

To test the robustness of the methods to noise we add Gaussian noise to each of the signal samples utilized in each method and then measure the contamination of the recovered Fourier series coefficients for (n, B)-block sparse functions $f\colon [0, 2\pi] \to \mathbb{C}$ with bandwidth $N = 2^{22}$, number of blocks $n = 3$, and block length $B = 2^4$. More specifically, each method considered herein utilizes a set of samples from f given by $\mathbf{f} = (f(x_j))_{j=0}^{m-1}$ for

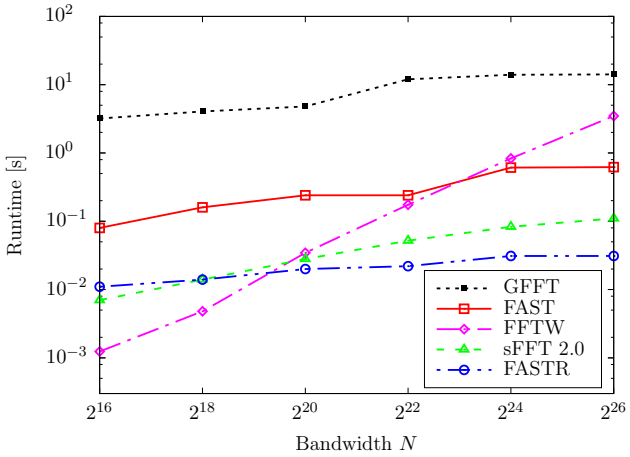


Figure 3: Runtime comparison for $n = 2$ blocks of length $B = 64$.

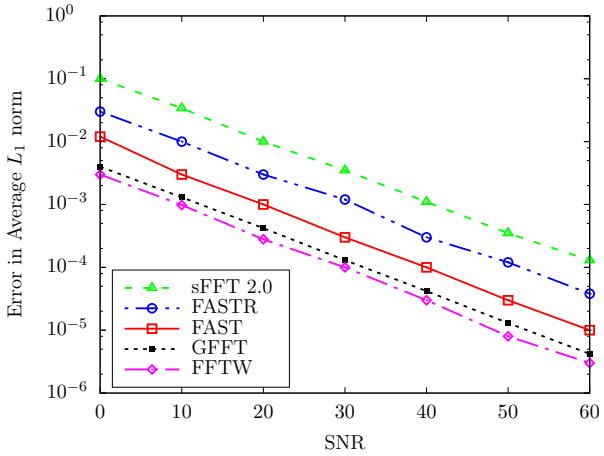


Figure 4: Robustness to noise for bandwidth $N = 2^{22}$ and $n = 3$ blocks of length $B = 2^4$.

some $x_0, \ldots, x_{m-1} \in [0, 2\pi)$ with $m \leq N$. For the experiments in this section we instead provide each algorithm with noisy function evaluations of the form $(f(x_j) + n_j)_{j=0}^{m-1}$, where each $n_j \in \mathbb{C}$ is a complex Gaussian random variable with mean 0. The n_j are then rescaled so that the total additive noise $\mathbf{n} = (n_j)_{j=0}^{m-1}$ achieves the signal-to-noise ratios (SNRs) considered in Figure 4.[7]

Recall that the two randomized algorithms compared herein (SFT 2.0 and FASTR) are both tuned to guarantee exact recover of block sparse functions with probability at least 0.9 in all experiments. For our noise robustness experiments this ensures that the correct frequency support, S, is found for at least 90 of the 100 trial signals used to generate each point plotted in Figure 4. All the other (deterministic) methods always find this correct support for all noise levels considered herein after sorting their output Fourier coefficient estimates by magnitude. Figure 4 plots the average ℓ^1-error over the true Fourier coefficients for frequencies in the correct frequency support S of each trial signal, averaged over the at least 90 trial runs at each point for which each sparse Fourier

[7]The SNR is defined to be $\text{SNR} = 20 \log\left(\frac{\|\mathbf{f}\|_2}{\|\mathbf{n}\|_2}\right)$, where $\mathbf{f}$ and $\mathbf{n}$ are as given above.

transform correctly identified S. More specifically, it graphs

$$\frac{1}{Bn}\sum_{\omega\in S}\left|c_\omega - x_\omega\right|,$$

where c_ω are the true Fourier coefficients for frequencies $\omega \in S$, and x_ω are their recovered approximations, averaged over the at least 90 trial signals where each method correctly identified S.

Looking at Figure 4 one can see that all of the Fourier transform algorithms in our experiments are robust to noise. Overall, however, the deterministic algorithms (FAST, GFFT and FFTW) are more robust than randomized algorithms (FASTR and sFFT 2.0). As expected, FFTW is the most robust algorithm in this experiment, followed closely by GFFT. For the randomized algorithms, FASTR is more robust than sFFT 2.0.

6 Conclusion

In this paper we developed the fastest known deterministic SFT method for the recovery of polynomially structured sparse input functions. However, there are still some remaining avenues for future research. To begin with one could try to find other types of structured sparsity that also guarantee an upper bound on the sparsity of the frequency restrictions for all possible residues. Considering a structure generated by polynomials was merely the most obvious choice, as polynomials naturally agree well with hashing modulo prime numbers and therefore interact well with the the number theoretic constructions used herein. One could also investigate whether utilizing structured sparsity might actually improve the runtimes of existing randomized SFT algorithms for unstructured sparsity.

It would also be interesting to know whether the results presented herein can be transferred to the non-periodic, continuous case, i.e., to sparse functions defined on the whole real line. Results in [6] about porting randomized SFT algorithms to the continuous setting suggest that this should be possible for a randomized version of Algorithm 1.

Acknowledgements

Sina Bittens was supported in part by the DFG in the framework of the GRK 2088. Mark Iwen and Ruochuan Zhang were both supported in part by NSF DMS-1416752. The authors would also like to thank both Felix Krahmer for introducing them at TUM in the summer of 2016, as well as Gerlind Plonka for her ongoing support, and particularly for her generosity in providing resources that aided in the writing of this paper.

References

[1] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. *COLT*, pages 381–393, 2010.

[2] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *FOCS*, 44:146–159, 2003.

[3] J. Bailey, M. A. Iwen, and C. V. Spencer. 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] S. Bittens. Sparse FFT for Functions with Short Frequency Support. *Dolomites Research Notes on Approximation*, 2017. To appear.

[5] L. I. Bluestein. A Linear Filtering Approach to the Computation of Discrete Fourier Transform. *Audio and Electroacoustics, IEEE Transactions on*, 18(4):451–455, 1970.

[6] P. Boufounos, V. Cevher, A. C. Gilbert, Y. Li, and M. J. Strauss. What's the frequency, Kenneth?: Sublinear Fourier sampling off the grid. *RANDOM/APPROX*, 2012.

[7] J. Bourgain, S. Dilworth, K. Ford, S. Konyagin, D. Kutzarova, et al. Explicit constructions of RIP matrices and related problems. *Duke Mathematical Journal*, 159(1):145–185, 2011.

[8] V. Cevher, M. Kapralov, J. Scarlett, and A. Zandieh. An Adaptive Sublinear-Time Block Sparse Fourier Transform. `http://arxiv.org/abs/1702.01286`, 2017.

[9] M. Cheraghchi and P. Indyk. Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 298–317. Society for Industrial and Applied Mathematics, 2016.

[10] A. Christlieb, D. Lawlor, and Y. Wang. A multiscale sub-linear time Fourier algorithm for noisy data. *Applied and Computational Harmonic Analysis*, 40(3):553–574, 2016.

[11] P. Feng and Y. Bresler. Spectrum-blind minimum-rate sampling and reconstruction of multiband signals. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 3, pages 1688–1691 vol. 3, May 1996.

[12] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*. Birkhäuser Basel, 2013.

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

[14] A. C. Gilbert, P. Indyk, M. A. Iwen, and L. Schmidt. Recent Developments in the Sparse Fourier Transform: A compressed Fourier transform for big data. *IEEE Signal Processing Magazine*, 31(5):91–100, 2014.

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

[16] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster GPS via the sparse Fourier transform. *MOBICOM*, 2012.

[17] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Near-optimal algorithm for sparse Fourier transform. *STOC*, 2012.

[18] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. sFFT: Sparse Fast Fourier Transform. `http://groups.csail.mit.edu/netmit/sFFT/`, 2012.

[19] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse Fourier transform. *SODA*, 2012.

[20] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi. Ghz-wide sensing and decoding using the sparse Fourier transform. *INFOCOM*, 2014.

[21] X. Hu, M. A. Iwen, and H. Kim. Rapidly computing sparse Legendre expansions via sparse Fourier transforms. *Numerical Algorithms*, pages 1–31, 2015.

[22] P. Indyk, M. Kapralov, and E. Price. (Nearly) sample-optimal sparse Fourier transform. *SODA*, 2014.

[23] M. A. Iwen. Combinatorial Sublinear-Time Fourier Algorithms. *Foundations of Computational Mathematics*, 10:303–338, 2010.

[24] M. A. Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 34:57–82, 2013.

[25] M. A. Iwen. MSU's Sparse Fourier Repository. `http://sourceforge.net/projects/aafftannarborfa/`, 2013.

[26] 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, 2007.

[27] M. A. Iwen and C. V. Spencer. Improved Bounds for a Deterministic Sublinear-Time Sparse Fourier Algorithm. *Conference on Information Systems (CISS)*, 2008.

[28] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer New York, 2005.

[29] J. Laska, S. Kirolos, Y. Massoud, R. Baraniuk, A. C. Gilbert, M. A. Iwen, and M. J. Strauss. Random sampling for analog-to-information conversion of wideband signals. In *Design, Applications, Integration and Software, 2006 IEEE Dallas/CAS Workshop on*, pages 119–122. IEEE, 2006.

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

[31] S. Merhi, R. Zhang, M. A. Iwen, and A. Christlieb. A New Class of Fully Discrete Sparse Fourier Transforms: Faster Stable Implementations with Guarantees. *preprint*, 2017.

[32] M. Mishali and Y. C. Eldar. Blind multiband signal reconstruction: Compressed sensing for analog signals. *IEEE Transactions on Signal Processing*, 57(3):993–1009, March 2009.

[33] M. Mishali and Y. C. Eldar. From Theory to Practice: Sub-Nyquist Sampling of Sparse Wideband Analog Signals. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):375–391, April 2010.

[34] M. Mishali, Y. C. Eldar, O. Dounaevsky, and E. Shoshan. Xampling: Analog to digital at sub-Nyquist rates. *IET Circuits, Devices Systems*, 5(1):8–20, January 2011.

[35] M. Mishali, Y. C. Eldar, and J. A. Tropp. Efficient sampling of sparse wideband analog signals. In *2008 IEEE 25th Convention of Electrical and Electronics Engineers in Israel*, pages 290–294, Dec 2008.

[36] H. Montgomery and R. Vaughan. *Multiplicative Number Theory I: Classical Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2007.

[37] L. Morotti. Explicit universal sampling sets in finite vector spaces. *Applied and Computational Harmonic Analysis*, 2016.

[38] G. Plonka and K. Wannenwetsch. A deterministic sparse FFT algorithm for vectors with small support. *Numerical Algorithms*, 71(4):889–905, 2016.

[39] G. Plonka and K. Wannenwetsch. A sparse fast Fourier algorithm for real nonnegative vectors. *Journal of Computational and Applied Mathematics*, 321:532 – 539, 2017.

[40] L. R. Rabiner, R. W. Schafer, and C. M. Rader. The Chirp-z Transform Algorithm. *IEEE Transactions on Audio and Electroacoustics*, 17:86–92, 1969.

[41] B. Segal and M. A. Iwen. Improved sparse Fourier approximation results: faster implementations and stronger guarantees. *Numerical Algorithms*, 63(2):239–263, 2013.

[42] P. Yenduri and A. C. Gilbert. Compressive, collaborative spectrum sensing for wideband cognitive radios. *ISWCS*, pages 531–535, 2012.

[43] P. K. Yenduri, A. Z. Rocca, A. S. Rao, S. Naraghi, M. P. Flynn, and A. C. Gilbert. A low-power compressive sampling time-based analog-to-digital converter. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(3):502–515, 2012.