PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Special issue dedicated to Annie Cuyt on the occasion of her 60th birthday, Volume 10 · 2017 · Pages 43–55

Sparse FFT for Functions with Short Frequency Support

Sina Bittens[a]

Communicated by A. Cuyt and W. S. Lee

Abstract

In this paper we derive a deterministic fast discrete Fourier transform algorithm for a 2π-periodic function f whose Fourier coefficients with significantly large magnitude are contained inside a support interval of length B. The algorithm is based on a method for the efficient recovery of B-sparse 2π-periodic functions presented by Iwen in [15]. If a good bound B on the support length is known apriori, our algorithm needs $\mathcal{O}\left(B\log B\frac{\log^2 N/B}{\log\log N/B}\right)$ arithmetical operations while using $\mathcal{O}\left(B\frac{\log^2 N/B}{\log\log N/B}\right)$ samples of the input function f.

Keywords. discrete Fourier transform, deterministic sparse FFT, sublinear sparse FFT, Chinese Remainder Theorem
AMS Subject Classification. 65T50, 42A16, 94A12, 11A07

1 Introduction

Algorithms for the fast Fourier transform (FFT) are of great importance in numerical mathematics, especially in signal and image processing, but for an arbitrary vector $\mathbf{x}\in\mathbb{C}^N$ the effort of $\mathcal{O}(N\log N)$ arithmetical operations for computing the discrete Fourier transform (DFT) $\widehat{\mathbf{x}}$ is optimal (see [21]). In the case of a sparse input vector $\mathbf{x}$, meaning that only a few of its entries have a significantly large absolute value, research done in recent years showed that it is possible to develop faster, sublinear-time DFT algorithms. Many of the presented algorithms (see, e.g., [3, 9, 11, 20, 13, 12, 22]) rely on randomization techniques where the returned vector is a good approximation with a constant probability less than 1. With this usually tunable, small probability of failure the fastest of these algorithms estimate the Fourier transform of B-sparse input vectors or functions in only $\mathcal{O}(B\log N)$ time (see, e.g., [12]). Setting $B=N^\delta$ for some $0<\delta<1$, the algorithm in [22] achieves a runtime of $\mathcal{O}(B\log B)$, where $\log N=\mathcal{O}(\log B)$. More details about such techniques can be found in a recent survey [10].

Deterministic sparse FFT methods have also been found, for example algorithms modifying Prony's method (see, e.g., [14, 23, 27]), which have complexities of $\mathcal{O}(B^3)$. However, many Prony-based methods are numerically unstable, as noisy data leads to a non-singular Hankel system, which causes perturbation of the eigenvectors corresponding to small eigenvalues, and hence perturbed zeroes of the Prony polynomial. Other deterministic FFT algorithms use arithmetic progressions [1, 2] or the Chinese Remainder Theorem [15, 16].

In the case of a short support, where all significant entries of the input vector are contained in a support interval of length B, it is possible to obtain a deterministic algorithm with runtime $\mathcal{O}(B\log N)$ using periodized vectors [24]. If the input vector has real non-negative entries, similar methods allow an algorithm that needs $\mathcal{O}(B\log B\log(N/B))$ arithmetic operations without requiring the apriori knowledge of a good bound on the support length B [25].

The setting of vectors with short support is, for example, of importance in computer tomography reconstructions and X-ray microscopy, where the compact support is often known apriori. In our paper we also focus on this case, and will derive an algorithm for functions with a short frequency support of length B. The approach is based on the sublinear-time method for general B-sparse functions presented by Iwen in [15], which has a runtime of $\mathcal{O}\left(B^2\cdot\frac{\log^4 N\log^2(B\log N)}{\log^2 B\log\log N/B}\right)$ and uses $\mathcal{O}\left(B^2\cdot\frac{\log^4 N\log(B\log N)}{\log^2 B\log\log N/B}\right)$ samples. It relies on reconstructing the energetic frequencies from their residues modulo certain integers with the Chinese Remainder Theorem, where the residues can be obtained from the DFT of several vectors of equispaced function samples with lengths that are small compared to the bandwidth of the function.

If the input function f has a frequency support length that is bounded by an apriori known constant B, the technique introduced in [15] can be strongly simplified and leads to a new algorithm with complexity $\mathcal{O}\left(B\log B\cdot\frac{\log^2 N/B}{\log\log N/B}\right)$, where only $\mathcal{O}\left(B\frac{\log^2 N/B}{\log\log N/B}\right)$ samples of f are being used. One can clearly see that our technique is faster and uses fewer samples; its runtime and sparsity even scale sub-quadratically in the support length B, which is also the sparsity. A support length $B=\lceil N^{1/4}\rceil$, for example, yields a still-sublinear runtime of $\mathcal{O}\left(B^2\frac{\log^4 B}{\log\log B}\right)$ for the algorithm in [15], but only of $\mathcal{O}\left(B\frac{\log^3 B}{\log\log B}\right)$ for our method.

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

Choosing $B = \lceil \sqrt{N} \rceil$, for which the algorithm in [15] is not sublinear anymore, we obtain a runtime of $\mathcal{O}\left(B \frac{\log^3 B}{\log \log B}\right)$, which only differs by a factor of $\frac{\log^2 B}{\log \log B}$ from the runtime of $\mathcal{O}(B \log B)$ of the algorithm in [24].

This paper is structured as follows. First, we fix the notation and recall the aspects of Algorithm 2 in [15] that are relevant for our setting in §2. In §3 we introduce the simplified algorithm for functions with short frequency support and analyze its runtime and sampling bounds. Finally, in §4, we give some numerical results and compare the algorithm to the algorithm for vectors with short support presented in [24].

2 Preliminaries and Required Technical Background

2.1 Preliminaries

Throughout this paper let $B \leq N \in \mathbb{N}$. We always consider a 2π-periodic function

$$f : [0, 2\pi] \to \mathbb{C}, \quad f(x) = \sum_{\omega=-\lceil \frac{N}{2} \rceil+1}^{\lfloor \frac{N}{2} \rfloor} c_\omega e^{i\omega x}$$

with bandwidth N and *finite Fourier transform*

$$\mathbf{c}(f) := (c_\omega)_{\omega=-\lceil \frac{N}{2} \rceil+1}^{\lfloor \frac{N}{2} \rfloor}.$$

We say that a frequency ω is *energetic* if the corresponding Fourier coefficient c_ω has a *significantly large* absolute value,

$$|c_\omega| > \varepsilon$$

for some threshold parameter $\varepsilon > 0$. A function f is called *B-sparse* if it has only B energetic frequencies. If all energetic frequencies are contained in a support interval

$$\{\omega_1, \omega_1 + 1, \ldots, \omega_1 + B - 1\} \subset \left\{ -\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \right\}$$

of length B, we say that f has *short frequency support of length B*, and we can write

$$f(x) = \sum_{b=0}^{B-1} c_{\omega_1+b} \cdot e^{i(\omega_1+b)x}.$$

Here, we do not require all frequencies to be energetic; some of the Fourier coefficients may even be zero. We recall the definition of the *discrete Fourier transform (DFT)* of length M, which maps $\mathbf{A} = (A(j))_{j=0}^{M-1} \in \mathbb{C}^M$ to $\widehat{\mathbf{A}} = (\widehat{A}(\omega))_{\omega=-\lceil \frac{M}{2} \rceil+1}^{\lfloor \frac{M}{2} \rfloor} \in \mathbb{C}^M$ via

$$\widehat{\mathbf{A}}(\omega) := (\tilde{\mathbf{F}}_M \cdot \mathbf{A})(\omega) := \frac{1}{M} \cdot \sum_{j=0}^{M-1} e^{\frac{-2\pi i j \omega}{M}} \cdot A(j) \quad \forall \omega \in \left\{ -\left\lceil \frac{M}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{M}{2} \right\rfloor \right\},$$

where $\tilde{\mathbf{F}}_M$ is the *centered M-th Fourier matrix*

$$\tilde{\mathbf{F}}_M := \frac{1}{M} \cdot \left({\omega_M}^{j\omega} \right)_{\omega=-\lceil \frac{M}{2} \rceil+1, j=0}^{\lfloor \frac{M}{2} \rfloor, M-1} = \frac{1}{M} \cdot \left({\omega_M}^{j(-\lceil \frac{M}{2} \rceil+1)} \cdot {\omega_M}^{j\omega} \right)_{\omega,j=0}^{M-1}$$

and $\omega_M := e^{\frac{-2\pi i}{M}}$ is the *M-th primitive root of unity*.

This version of the DFT allows us to consider a frequency range $\left\{ -\lceil \frac{N}{2} \rceil + 1, \ldots, \lfloor \frac{N}{2} \rfloor \right\}$ centered around 0 instead of the usual non-negative frequencies $\{0, \ldots, N-1\}$. We will always assume that we can evaluate the input function f at all necessary points, but that the occurring frequencies and the Fourier coefficients are not known. For any $M \in \mathbb{N}$, we denote by $\mathbf{A}_M$ the M-length vector consisting of equidistant samples of f,

$$\mathbf{A}_M := (A_M(j))_{j=0}^{M-1} := \left(f\left(\frac{2\pi j}{M} \right) \right)_{j=0}^{M-1}. \tag{1}$$

The basic idea of the algorithm is to apply the DFT to several vectors of this type with lengths that are small compared to the bandwidth N and to thus obtain a method with a runtime that is faster than the $\mathcal{O}(N \log N)$ runtime of the FFT.

From now on we denote by p_l, $l \in \mathbb{N}$, the *l-th natural prime number*. Additionally, we let $p_0 = 1$ for inductive purposes.

2.2 Reconstruction Procedure for One Frequency

We begin by sketching the ideas of Algorithm 2 in [15] that are required to reconstruct a 2π-periodic input function $f : [0, 2\pi] \to \mathbb{C}$ with short frequency support whose length is bounded by B. First we consider a function with only a single energetic frequency $\omega \in \{-\lceil \frac{N}{2} \rceil + 1, \dots, \lfloor \frac{N}{2} \rfloor\}$ and its Fourier coefficient $c_\omega \in \mathbb{C}\backslash\{0\}$,

$$f(x) = c_\omega \cdot e^{i\omega x}.$$

Thus, f is completely determined if we can reconstruct ω and c_ω. Applying the DFT to the vector $\mathbf{A}_M$ of $M \ll N$ equidistant samples of f yields that

$$\begin{aligned}\widehat{A_M}(l) &= \frac{1}{M}\sum_{j=0}^{M-1} e^{\frac{-2\pi i j l}{M}} \cdot c_\omega e^{\frac{2\pi i j \omega}{M}} = \frac{c_\omega}{M}\sum_{j=0}^{M-1} \omega_M^{j(l-\omega)} \\ &= \begin{cases} c_\omega, & \text{if } l \equiv \omega \bmod M, \\ 0, & \text{otherwise,} \end{cases} \quad \forall l \in \left\{-\left\lceil \frac{M}{2} \right\rceil + 1, \dots, \left\lfloor \frac{M}{2} \right\rfloor\right\}. \end{aligned} \tag{2}$$

Hence, $\widehat{\mathbf{A}_M}$ has exactly one nonzero entry, namely the Fourier coefficient corresponding to ω, and its index is the residue of ω modulo M:

$$\widehat{A_M}(\omega \bmod M) = c_\omega = \frac{1}{2\pi}\int_0^{2\pi} c_\omega e^{i\omega x} \cdot e^{-i\omega x} dx = \widehat{f}(\omega).$$

Consequently, (2) gives us the value of the Fourier coefficient c_ω and the residue of ω modulo M without actually knowing either. If we compute the DFTs of several such vectors, we obtain a system of simultaneous congruencies which can, under certain conditions, be solved with the Chinese Remainder Theorem (see [18], ch. I, §4).

Theorem 2.1 (Chinese Remainder Theorem (CRT)). *Let $M_1, \dots, M_L$ be pairwise relatively prime integers and $N \le \prod_{l=1}^L M_l$. Then there exists a unique solution modulo N of the system of simultaneous congruencies*

$$\begin{aligned} x &\equiv r_1 \bmod M_1, \\ &\vdots \\ x &\equiv r_L \bmod M_L. \end{aligned}$$

The unique solution of such a system can be computed in $\mathcal{O}(\sum_{l=1}^L \log M_l)$ time.

Hence, if $\widehat{\mathbf{A}_M}$ is known for enough pairwise relatively prime moduli $M_l \ll N$, $l \in \{1, \dots, L\}$, with $N \le \prod_{l=1}^L M_l$, the CRT implies that we can uniquely recover the frequency ω. The corresponding Fourier coefficient c_ω is already given by (2) as

$$\widehat{A_{M_l}}(\omega \bmod M_l) = c_\omega \quad \forall l \in \{1, \dots, L\}.$$

This means that, instead of computing the DFT of length N of $\widehat{\mathbf{A}_N}$, it suffices to calculate L DFTs of length M_l and reconstruct ω from its residues modulo the M_l.

Example 2.1. Let $f : [0, 2\pi] \to \mathbb{C}$, $f(x) = e^{i \cdot 210x}$ with bandwidth $N = 1000$. According to the CRT ω will be uniquely determined modulo N by its residues modulo $10, 11$ and 13. Locating the nonzero entries of $\widehat{\mathbf{A}_{10}}, \widehat{\mathbf{A}_{11}}$ and $\widehat{\mathbf{A}_{13}}$, we find that

$$\begin{aligned} \widehat{A_{10}}(0) = 1 \quad &\Rightarrow \quad \omega \equiv 0 \bmod 10, \\ \widehat{A_{11}}(1) = 1 \quad &\Rightarrow \quad \omega \equiv 1 \bmod 11 \quad \text{and} \\ \widehat{A_{13}}(2) = 1 \quad &\Rightarrow \quad \omega \equiv 2 \bmod 13. \end{aligned}$$

Now we can recover ω from its residues,

$$\begin{aligned} & \omega \equiv 0 \bmod 10 && \Rightarrow \quad \omega = 10a \equiv 1 \bmod 11 \\ \Rightarrow \quad & a \equiv 10 \bmod 11 && \Rightarrow \quad a = 11b + 10 \\ \Rightarrow \quad & \omega = 10 \cdot (11b + 10) && \Rightarrow \quad \omega = 110b + 100 \equiv 2 \bmod 13 \\ \Rightarrow \quad & b \equiv 1 \bmod 13 && \Rightarrow \quad b = 13c + 1 \\ \Rightarrow \quad & \omega = 110 \cdot (13c + 1) + 100 && \Rightarrow \quad \omega \equiv 210 \bmod 1430 \end{aligned}$$

for some $a, b, c \in \mathbb{Z}$. Since we have $\omega \in \{-\lceil \frac{N}{2} \rceil + 1, \dots, \lfloor \frac{N}{2} \rfloor\} = \{-499, \dots, 500\}$, we find that $\omega = 210$. The Fourier coefficient c_ω is given by (2) as

$$c_\omega = \widehat{A_{10}}(0) = 1.$$

For this computation three DFTs of length 10, 11 and 13 were necessary; hence only 34 samples instead of $N = 1000$ samples of f were used. Moreover, as there also exist fast algorithms for DFTs of vectors of arbitrary length (see, e.g., [5, 26, 7]), the three DFTs can be computed in $\mathcal{O}(\sum_{l=1}^3 M_l \log M_l)$ time, instead of calculating one DFT with complexity $\mathcal{O}(N \log N)$. The frequency reconstruction needs $\mathcal{O}(\sum_{l=1}^3 \log M_l)$ arithmetic operations, which is insignificant compared to the costs of the computation of the DFTs.

However, as soon as the considered function has more than one energetic frequency, their residues can coincide modulo various integers, which means that in the case of a B-sparse function the moduli M_l cannot be chosen arbitrarily.

2.3 Reconstruction Procedure for Several Frequencies

Let us now examine a 2π-periodic function $f : [0, 2\pi] \to \mathbb{C}$ with B distinct frequencies $\omega_1, \dots, \omega_B \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \dots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$ and their Fourier coefficients $c_{\omega_1}, \dots, c_{\omega_B} \in \mathbb{C}\backslash\{0\}$,

$$f(x) = \sum_{j=1}^{B} c_{\omega_j} \cdot e^{i\omega_j x}.$$

Since all frequencies are distinct, there exists an $s \in \mathbb{N}$ such that the frequency residues modulo s do not coincide. This motivated the use of the notion of separation in [15].

Definition 2.1 (Separation)**.** Let s and $B \in \mathbb{N}$ and $\omega_1, \dots, \omega_B \in \mathbb{Z}$ be distinct. Then s *separates* the integers $\omega_1, \dots, \omega_B$ if

$$\omega_j \bmod s \neq \omega_l \bmod s \quad \forall j, l \in \{1, \dots, B\},\ j \neq l.$$

We assume for the moment that such an s is known. If we apply the DFT to the vector $\mathbf{A}_s$ of s equidistant samples of f from (1), we find with the linearity of the DFT that

$$\widehat{A_s}(l) = \begin{cases} c_{\omega_1}, & \text{if } l \equiv \omega_1 \bmod s, \\ \vdots & \vdots \\ c_{\omega_B}, & \text{if } l \equiv \omega_B \bmod s, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall l \in \left\{-\left\lceil \frac{s}{2} \right\rceil + 1, \dots, \left\lfloor \frac{s}{2} \right\rfloor\right\}. \tag{3}$$

Hence, $\widehat{\mathbf{A}_s}$ has exactly B nonzero entries and their indices are the residues of the ω_j modulo s, as they cannot coincide due to the separation property of s. However, we also obtain from (3) that, if the Fourier coefficients of some of the frequencies are equal, we cannot uniquely match the residues modulo s to the frequencies.

In order to apply the CRT for the reconstruction of $\omega_1, \dots, \omega_B$, we have to choose the moduli in a way that allows us to efficiently compute the residues of the frequencies. For this we can use that for all $p \in \mathbb{N}$ and $a, b \in \mathbb{Z}$ it holds that

$$(a \equiv b \bmod ps \quad \Rightarrow \quad a \equiv b \bmod s) \quad \Leftrightarrow \quad (a \not\equiv b \bmod ps \quad \Leftarrow \quad a \not\equiv b \bmod s). \tag{4}$$

Consequently, if s separates the frequencies $\omega_1, \dots, \omega_B$, so does ps for all $p \in \mathbb{N}$, which means that we can generate infinitely many separating natural numbers. However, the ps are of course not pairwise relatively prime anymore. Instead, we use the residues modulo the p, choosing them such that the CRT holds for them and s. Assuming that the residues modulo ps are known, we have

$$a \bmod p = (a \bmod ps) \bmod p \quad \forall a \in \mathbb{Z}, \tag{5}$$

and can recover the frequencies from their residues modulo the p and s. The simplest way to ensure the prerequisites of the CRT is to take the L smallest prime numbers $t_1, \dots, t_L$ that are relatively prime to s such that

$$s \cdot \prod_{l=1}^{L-1} t_l < N \leq s \cdot \prod_{l=1}^{L} t_l, \quad \gcd(t_l, s) = 1 \quad \forall l \in \{1, \dots, L\}.$$

We show now how the residues modulo $t_l s$ can be computed efficiently. In order to simplify the notation, we just consider an arbitrary prime p and the frequency ω_1, but the same procedure works for all primes t_l and all energetic ω_j. From (3) we know that

$$\widehat{A_{ps}}(\omega_1 \bmod ps) = c_{\omega_1} = \widehat{A_s}(\omega_1 \bmod s),$$

so the residue of ω_1 modulo ps can be found by comparison with its residue modulo s. Set $r_0 := \omega_1 \bmod s$. Then ω_1 is of the form

$$\omega_1 = r_0 + a \cdot s$$

for an $a \in \mathbb{Z}$, and its residue modulo ps satisfies

$$\omega_1 \bmod ps = (r_0 + as) \bmod ps = r_0 + (a \bmod p) \cdot s =: r_0 + b_{\min} \cdot s \tag{6}$$

for $b_{\min} := a \bmod p \in \{0, \dots, p-1\}$. Thus there are only p possibilities for the residue of ω_1 modulo ps. Recall that due to the separation property of s we have

$$\omega_1 \bmod s \neq \omega_j \bmod s \quad \forall j \in \{2, \dots, B\}.$$

Then

$$(\omega_1 + bs) \bmod s \neq \omega_j \bmod s \quad \forall j \in \{2, \dots, B\},$$

and therefore (4) yields that

$$(\omega_1 + bs) \bmod ps \neq \omega_j \bmod ps \quad \forall j \in \{2, \dots, B\}$$

for all $b \in \{0, \dots, p-1\}$. Consequently, none of the p possible values $\omega_1 + bs$ for the residue of ω_1 modulo ps from (6) can coincide with the residue of another energetic frequency ω_j modulo ps for $j \neq 1$. Exactly one of the p values $\widehat{A_{ps}}(r_0 + bs)$, where

$b \in \{0,\dots,p-1\}$, is not zero but equal to $\widehat{A_s}(r_0) = c_{\omega_1}$; hence we can determine $\omega_1 \mod ps$ by comparing $\widehat{A_s}(r_0)$ and $\widehat{A_{ps}}(r_0+bs)$ for all possible values of b,

$$\begin{aligned} & \omega_1 \mod ps = r_0 + b_{\min}\cdot s \\ \Leftrightarrow \quad & \left|\widehat{A_s}(r_0) - \widehat{A_{ps}}(r_0 + b_{\min}\cdot s)\right| = \min_{b=\{0,\dots,p-1\}} \left|\widehat{A_s}(r_0) - \widehat{A_{ps}}(r_0+bs)\right|. \end{aligned} \tag{7}$$

Having found the residue of ω_1 modulo ps from (7), its residue modulo p can be calculated with the help of (5). After computing the residues modulo all t_l, we can uniquely reconstruct ω_1 from its residues modulo $s, t_1,\dots,t_L$. Since s separates all occurring frequencies, the Fourier coefficient c_{ω_1} is given by (3) as

$$c_{\omega_1} = \widehat{A_s}(\omega_1 \mod s).$$

The remaining frequencies and their coefficients can be found analogously.

3 Recovery of Functions with Short Frequency Support

So far we just assumed that an s that separates all energetic frequencies is known. However, for arbitrary B-sparse functions, as in [15], this is impossible. With some combinatorial constructions one can guarantee that at least more than half of K integers $s_1,\dots,s_K$, satisfying certain additional properties, separate all energetic frequencies, and apply median techniques to find the correct frequencies and coefficient estimates. In the case of functions with short frequency support, though, this approach can be simplified.

3.1 Algorithm for Functions with Short Frequency Support

Having recalled the ideas of Algorithm 2 in [15] that are necessary for our setting, we can now consider functions of the form

$$f(x) = \sum_{j=0}^{B-1} c_{\omega_1+j}\cdot e^{i(\omega_1+j)x},$$

where all energetic frequencies are contained in the B-length interval $\{\omega_1,\dots,\omega_1+B-1\}$. Then we already know that B separates all of them. In the reconstruction procedure outlined in §2.3 the moduli we used were the L smallest primes t_l that do not divide B with $N \le B\cdot\prod_{l=1}^{L} t_l$. To simplify the estimation of the runtime and sample bounds, and to avoid collision with the t_l, we set s as the smallest power of 2 that is greater than B,

$$s := 2^\alpha, \quad \text{where} \quad \alpha := \lfloor \log_2 B \rfloor + 1.$$

Then $t_1,\dots,t_L$ can be chosen as the L smallest odd primes satisfying

$$B\cdot\prod_{l=1}^{L-1} t_l < N \le B\cdot\prod_{l=1}^{L} t_l, \tag{8}$$

and, for inductive purposes, we set $t_0 := 1$. As s is greater than B, it still separates the B energetic frequencies $\omega_1, \omega_1+1,\dots,\omega_1+B-1$. Furthermore, s is relatively prime to all small odd primes $t_1,\dots,t_L$, so we can indeed uniquely recover the frequencies from their residues modulo $s, t_1,\dots,t_L$.

Remark 1. Using the reconstruction procedure with s is just applying Algorithm 2 in [15] to the first element of a B-majority selective collection of sets, $\mathscr{S}$, whose elements do not have to be primes (see [15], §3). The combinatorial considerations necessary for the general case of B energetic frequencies are rendered redundant by the fact that for functions with short frequency support of length B we always find an $s = 2^\alpha > B$ that separates all energetic frequencies, so we do not require s to separate any B-element subset of $\left\{-\left\lceil \frac{N}{2}\right\rceil+1,\dots,\left\lfloor\frac{N}{2}\right\rfloor\right\}$ anymore.

Due to the block structure of the energetic frequencies it is enough to perform the reconstruction procedure for a single energetic frequency $\tilde{\omega}$ and find the remaining ones by examining whether the absolute values of the Fourier coefficients of the $2B-1$ frequencies in $\{\tilde{\omega}-B+1, \tilde{\omega}-B+2,\dots,\tilde{\omega}+B-1\}$ are significantly large. All of the at most B energetic frequencies have to be contained in this set, as the distance of any energetic frequency ω to $\tilde{\omega}$ can be at most $B-1$. The Fourier coefficients are given without any further computation, since they are just the significantly large entries of $\widehat{\mathbf{A}_{t_l s}}$. The $2B-1$ possibly energetic, consecutive frequencies will not be distinct modulo s, but they are separated by $t_1 s = 3s$, so their Fourier coefficients are given by (3) as

$$c_\omega = \widehat{A_{3s}}(\omega \mod 3s)$$

for all $\omega \in \{\tilde{\omega}-B+1,\dots,\tilde{\omega}+B-1\}$. The frequency $\tilde{\omega}$ can be obtained from the index of the largest magnitude entry and the residues modulo the t_l, as the indices of all significantly large entries of $\widehat{\mathbf{A}_s}$ are the residues of the energetic frequencies modulo s.

We summarize this procedure in Algorithm 1, which finds the energetic frequencies and the Fourier coefficients of a function f with bandwidth N and short frequency support length bounded by B.

The function `extended_gcd` in line 16 finds the greatest common divisor g of two integers a and b using the extended Euclidean algorithm, as well as two integers u and v satisfying Bézout's identity,

$$g = \gcd(a,b) = ua + vb.$$

In Algorithm 1 we always have $g = 1$ by choice of s and $t_1,\dots,t_L$.

Algorithm 1 Algorithm for Functions with Short Frequency Support

Input: 2π-periodic function f, integers $B \leq N$, accuracy level ε.
Output: The set R of at most B energetic frequencies of f and the vector $\mathbf{x}$ of estimates for their Fourier coefficients.

Initialize $R \leftarrow \emptyset$
Find L and the smallest odd primes $t_1, \ldots, t_L$ s.t. $B \cdot \prod_{l=1}^{L-1} t_l < N \leq B \cdot \prod_{l=1}^{L} t_l$.
Set $s := 2^\alpha$, where $\alpha := \lfloor \log_2 B \rfloor + 1$.
for l from 0 to L **do**
    $\mathbf{A}_{t_l s} \leftarrow \left(f\left(\frac{2\pi j}{t_l s}\right)\right)_{j=0}^{t_l s - 1}$
    $\widehat{\mathbf{A}_{t_l s}} \leftarrow \mathbf{DFT}[\mathbf{A}_{t_l s}]$
end for

IDENTIFICATION OF ONE OF THE ENERGETIC FREQUENCIES

$r_0 \leftarrow \underset{j \in \{0,\ldots,s-1\}}{\operatorname{argmax}} \left|\widehat{A_s}(j)\right|$. ▷ $\tilde{\omega} \equiv r_0 \mod s$ for an energetic frequency $\tilde{\omega}$.
for l from 1 to L **do**
    $b_{\min} \leftarrow \underset{b \in \{0,\ldots,t_l - 1\}}{\operatorname{argmin}} \left(\left| \widehat{A_s}(r_0) - \widehat{A_{t_l s}}(b \cdot s + r_0) \right| \right)$
    $r_l \leftarrow (b_{\min} \cdot s + r_0) \mod t_l$ ▷ $\tilde{\omega} \equiv r_l \mod t_l$.
end for

RECONSTRUCTION OF $\tilde{\omega}$ FROM ITS RESIDUES

Set $l = 0$, $\tilde{\omega} = r_0$ and $n = s$.
while $l < L$ **do**
    Set $l = l + 1$
    $(g, u, v) \leftarrow \texttt{extended\_gcd}(n, t_l)$ ▷ $g = 1$ by choice of $s, t_1, \ldots, t_L$.
    $\tilde{\omega} = n\,(((r_l - \tilde{\omega}) \cdot u) \mod t_l) + \tilde{\omega}$
    $n = t_l \cdot n$
end while
Set $\tilde{\omega} = \tilde{\omega} \mod n$ and shift $\tilde{\omega}$ into the range $\left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor\right\}$, since $N \leq n$.

IDENTIFICATION OF THE REMAINING FREQUENCIES AND COEFFICIENTS

for ω from $\tilde{\omega} - B + 1$ to $\tilde{\omega} + B - 1$ **do**
    if $|\widehat{A_{t_1 s}}(\omega \mod t_1 s)| > \varepsilon$ **then**
        $R \leftarrow R \cup \{\omega\}$
        $\mathbf{x}(\omega) \leftarrow \widehat{A_{3s}}(\omega \mod 3s)$
    end if
end for

3.2 Runtime and Sample Bounds

Proving runtime and sample bounds for Algorithm 1 requires some preliminary results about the occurring sums of prime numbers. Recall that we have to compute DFTs of length $t_l s$ for all $l \in \{0,\dots,L\}$, so we have to estimate the number of necessary samples,

$$\sum_{l=0}^{L} t_l s, \tag{9}$$

of the input function, and the runtime of the computation of the DFTs,

$$\mathcal{O}\left(\sum_{l=0}^{L} t_l s \log(t_l s)\right) = \mathcal{O}\left(s\sum_{l=0}^{L} t_l \log t_l + s\log s \sum_{l=0}^{L} t_l\right). \tag{10}$$

First we estimate the largest of the small odd primes, t_L. The following result about the smallest M primes $p_1,\dots,p_M$, including $p_1 = 2$, has been shown in [17], Lemma 4.

Lemma 3.1. *Denote by p_l the l-th prime and let $B \le N \in \mathbb{N}$. If $M \in \mathbb{N}$ satisfies*

$$\prod_{l=1}^{M-1} p_l < \frac{N}{B} \le \prod_{l=1}^{M} p_l,$$

there exists a constant $a > 0$ with

$$p_M = \log\frac{N}{B} + \mathcal{O}\left(\frac{\log\frac{N}{B}}{\exp\left(a\sqrt{\log\log\frac{N}{B}}\right)}\right).$$

Lemma 3.2. *Denote by p_l the l-th prime and by t_l the l-th odd prime. Let $B \le N \in \mathbb{N}$. If $L \in \mathbb{N}$ satisfies*

$$\prod_{l=1}^{L-1} t_l < \frac{N}{B} \le \prod_{l=1}^{L} t_l,$$

there exists a constant $a > 0$ with

$$t_L = \log\frac{2N}{B} + \mathcal{O}\left(\frac{\log\frac{2N}{B}}{\exp\left(a\sqrt{\log\log\frac{2N}{B}}\right)}\right).$$

Proof. We have that

$$\prod_{l=1}^{L-1} t_l < \frac{N}{B} \le \prod_{l=1}^{L} t_l \quad\Leftrightarrow\quad \prod_{l=1}^{L} p_l < \frac{2N}{B} \le \prod_{l=1}^{L+1} p_l.$$

By Lemma 3.1 there exists an $a > 0$ such that

$$t_L = p_{L+1} = \log\frac{2N}{B} + \mathcal{O}\left(\frac{\log\frac{2N}{B}}{\exp\left(a\sqrt{\log\log\frac{2N}{B}}\right)}\right) = \mathcal{O}\left(\log\frac{N}{B}\right).$$

□

Further, we recall Lemma 5 and 6 in [17] about general sums of prime numbers.

Lemma 3.3. *For all $R \in \mathbb{N}$ it holds that*

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

and

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

We can now estimate the sums in (9) and (10).

Lemma 3.4. *Denote by t_l the l-th odd prime. Let $B \le N \in \mathbb{N}$ and $L \in \mathbb{N}$ such that*

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

Then we have

$$\sum_{l=0}^{L} t_l = \mathcal{O}\left(\frac{\log^2\frac{N}{B}}{\log\log\frac{N}{B}}\right) \quad \text{and} \quad \sum_{l=0}^{L} t_l \log t_l = \mathcal{O}\left(\log^2\frac{N}{B}\right).$$

Proof. For the first claim we apply the first result from Lemma 3.3 and find

$$\sum_{l=0}^{L} t_l \leq \sum_{\substack{p \leq t_L \\ p \text{ prime}}} p = \frac{{t_L}^2}{2 \log t_L} + \mathcal{O}\left(\frac{{t_L}^2}{\log^2 t_L}\right).$$

Using the estimate

$$t_L = \log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right), \quad \text{with} \quad A = \exp\left(a\sqrt{\log\log \frac{2N}{B}}\right),$$

from Lemma 3.2, we obtain

$$\begin{aligned}\sum_{l=0}^{L} t_l =& \frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{2\log\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)} + \mathcal{O}\left(\frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{\log^2\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)}\right) \\ =& \mathcal{O}\left(\frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right).\end{aligned}$$

For the second claim we employ the second result from Lemma 3.3, which yields

$$\begin{aligned}\sum_{l=0}^{L} t_l \log t_l \leq& \sum_{\substack{p \leq t_L \\ p \text{ prime}}} p \log p = \frac{{t_L}^2}{2} + \mathcal{O}\left(\frac{{t_L}^2}{\log t_L}\right) \\ =& \frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{2} + \mathcal{O}\left(\frac{\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)^2}{\log\left(\log \frac{2N}{B} + \mathcal{O}\left(\frac{\log \frac{2N}{B}}{A}\right)\right)}\right) \\ =& \mathcal{O}\left(\log^2 \frac{N}{B}\right).\end{aligned}$$

□

Combining all of these estimates, we can prove the following main result.

Theorem 3.5. *Let $B \leq N \in \mathbb{N}$, $\omega_1 \in \left\{-\left\lceil \frac{N}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{N}{2} \right\rfloor - B + 1\right\}$ and $f : [0, 2\pi] \to \mathbb{C}$ be a function with short frequency support length bounded by B and bandwidth N, i.e.,*

$$f(x) = \sum_{j=0}^{B-1} c_{\omega_1 + j} \cdot e^{i(\omega_1 + j)x}.$$

Then Algorithm 1 returns all energetic frequencies and Fourier coefficients of f in

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

time, and has a sampling complexity of

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

Proof. It is evident from the construction of Algorithm 1 that it returns the correct frequencies and Fourier coefficients, disregarding numerical errors. We can now calculate the runtimes of the different parts of Algorithm 1 using the observations made above.

The computational costs necessary to compute the small odd primes $t_1, \ldots, t_L$ can be disregarded. Even if we choose a bandwidth $N = 10^{10}$ and a support length $B = 1$, the largest required prime, $t_L = \mathcal{O}(\log \frac{N}{B})$, is 31. Usually one would consider greater support lengths, which means that even fewer t_l sufficed. Hence, the t_l can easily be found from precomputed lists of small primes in $\mathcal{O}(\log \frac{N}{B})$ time.

Since FFT algorithms with runtime $\mathcal{O}(M \log M)$ also exist for vectors of arbitrary length M (see [5, 26, 7]), the DFTs in lines 4 to 7 require

$$\mathcal{O}\left(\sum_{l=0}^{L} t_l s \log(t_l s)\right) = \mathcal{O}\left(s \sum_{l=0}^{L} t_l \log t_l + s \log s \sum_{l=0}^{L} t_l\right)$$

arithmetical operations. With Lemma 3.4 and $s = \mathcal{O}(B)$ we obtain

$$\begin{aligned}\mathcal{O}\left(\sum_{l=0}^{L} t_l s \log(t_l s)\right) =& \mathcal{O}\left(B \log^2 \frac{N}{B} + B \log B \cdot \frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right) \\ =& \mathcal{O}\left(B \log^2 \frac{N}{B}\left(1 + \frac{\log B}{\log\log \frac{N}{B}}\right)\right).\end{aligned}$$

Finding the largest magnitude entry of $\widehat{\mathbf{A}_s}$ in line 8 needs $\mathcal{O}(s)$ operations. The computation of the residues of an energetic frequency in lines 9 to 12 has a complexity of

$$\mathcal{O}\left(\sum_{l=0}^{L} t_l\right) = \mathcal{O}\left(\frac{\log^2 \frac{N}{B}}{\log\log \frac{N}{B}}\right).$$

The runtime of the extended Euclidean algorithm `extended_gcd`(n, t_l) in line 16 can be proven to be $\mathcal{O}(\log t_l)$, since $t_l < n$ for all l (see [6], ch. 31.2). Therefore lines 13 to 20 require $\mathcal{O}(\sum_{l=1}^{L} \log t_l)$ operations, which is insignificant compared to the runtime of lines 9 to 12. Finally, identifying the remaining frequencies in lines 21 to 26 has a runtime of $\mathcal{O}(B)$.

Combining all these results yields an overall arithmetical complexity of

$$\begin{aligned}
&\mathcal{O}\left(\sum_{l=0}^{L} t_l s \log(t_l s)\right) + \mathcal{O}(s) + \mathcal{O}\left(\sum_{l=0}^{L} t_l\right) + \mathcal{O}(B) \\
=&\mathcal{O}\left(B\log^2\frac{N}{B}\left(1+\frac{\log B}{\log\log\frac{N}{B}}\right) + B + \log\frac{N}{B} + B\right) \\
=&\mathcal{O}\left(B\log B\cdot\frac{\log^2\frac{N}{B}}{\log\log\frac{N}{B}}\right).
\end{aligned}$$

Further, we obtain with Lemma 3.4 that Algorithm 1 has a sampling complexity of

$$s\sum_{l=0}^{L} t_l = \mathcal{O}\left(B\cdot\frac{\log^2\frac{N}{B}}{\log\log\frac{N}{B}}\right).$$

□

4 Numerical Results

Hereafter, we present some numerical results regarding the runtime of Algorithm 1 and its performance for noisy input data. Due to the fact that we find the energetic frequency $\tilde{\omega}$ in line 8 by choosing the index of the element of $\widehat{\mathbf{A}_s}$ that has the largest absolute value, we always correctly identify an interval of length $2B-1$ centered around $\tilde{\omega}$ that contains the frequency support, as long as the noise does not dominate the original signal. Hence, our algorithm is still stable for noisy input data.

Additionally, we will compare it to the deterministic sparse inverse FFT algorithm for noisy input data presented by Plonka and Wannenwetsch in [24] (Algorithm 2), which recovers a vector $\mathbf{x} \in \mathbb{C}^{2^J}$ with short support length B from its Fourier transform $\widehat{\mathbf{x}}$ by considering periodizations $\mathbf{x}^{(j)} \in \mathbb{C}^{2^j}$ of $\mathbf{x}$,

$$\mathbf{x}^{(j)} = \left(\sum_{l=0}^{2^{J-j}-1} x_{k+2^j l}\right)_{k=0}^{2^j-1} \quad \text{for } j \in \left\{\lceil \log_2 B\rceil + 1, \ldots, J\right\}.$$

For noisy input data this can be achieved with an arithmetical complexity of $\mathcal{O}(B\log N)$, where $\mathcal{O}(B+\log N)$ samples of the input vector are being used. However, their approach requires that the length of the input vector $\mathbf{x}$ is a power of 2, whereas our algorithm works for arbitrary bandwidths N of f. In order to be able to compare these two algorithms, we always consider bandwidths that are of the form $N = 2^J$ in the following numerical experiments. Both algorithms have been implemented in MATLAB, and the code is freely available in [4, 28].

Further, we want to compare our method to MATLAB's `fft` function, which is a fast and highly optimized implementation of the fast Fourier transform, based on the FFTW library (see [19, 8]).

For sake of completeness we also compute the average runtimes of Algorithm 2 in [15] from numeric experiments for the short support lengths $B = 10$ and $B = 100$. As our method is essentially a simplification of that algorithm, using $L+1$ DFTs of length $t_l s$ each instead of $K(L+1)$ DFTs of length $\tilde{t}_l \tilde{s}_k$, where $t_l \approx \tilde{t}_l$, and the separating primes satisfy $\tilde{s}_1 = \mathcal{O}(s)$ and $s_1 < \cdots < s_K$, it will always be significantly faster than Algorithm 2 in [15], which is also evident by consideration of their theoretical runtimes of

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

of Algorithm 1 and

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

of Algorithm 2 in [15].

Figure 1 shows the average runtimes of Algorithm 1, Algorithm 2 introduced by Plonka and Wannenwetsch in [24], Algorithm 2 presented by Iwen in [15] and MATLAB's `fft` for 100 random input functions or vectors, where the absolute values of the real and imaginary parts of the Fourier coefficients are bounded by 10.

Of course any comparison of the implementation of the first three algorithms with the highly optimized implementation `fft` of the FFT must be flawed; however, we note that Algorithm 1 is much faster than both `fft` and Algorithm 2 in [24] for support

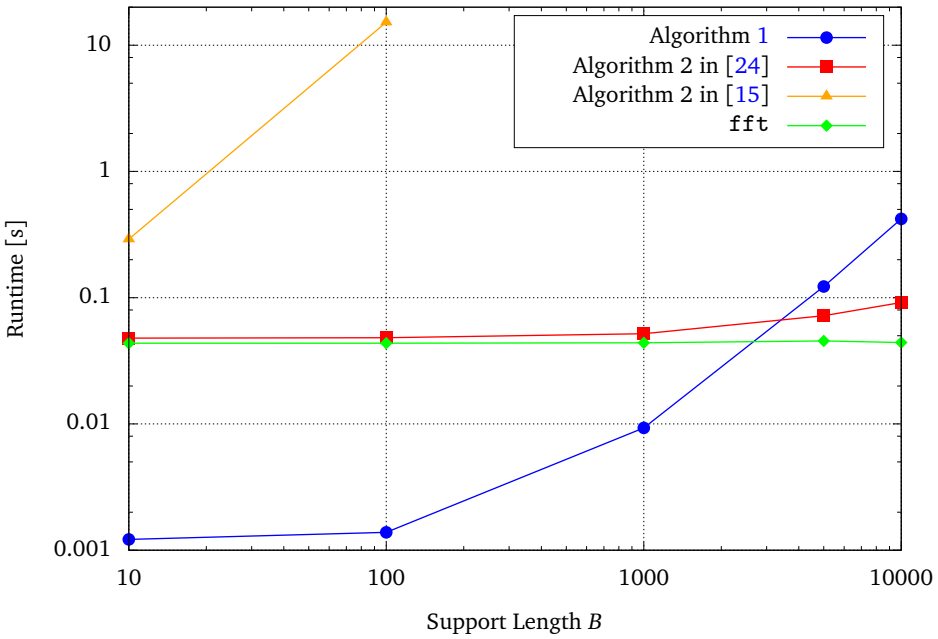

Figure 1: Average runtimes of Algorithm 1, Algorithm 2 in [24], Algorithm 2 in [15] and MATLAB's `fft` for 100 random input functions with support length B, bandwidth $N = 2^{20}$ and threshold $\varepsilon = 10^{-4}$.

lengths up to $B = 1000 \approx \sqrt{N}$ and that Algorithm 2 in [24] is as fast as `fft` for the same short support lengths. For greater support lengths both algorithms for functions with short frequency support perform much slower than `fft`, whose runtime $\mathcal{O}(N \log N)$ is independent of the support length, whereas the short support algorithms both have runtimes that are almost linear or linear in B. Additionally, it can be seen that the runtime of Algorithm 1 increases much faster in the support length B than the runtime of Algorithm 2 in [24]. As expected, even for a support of length $B = 100$ the runtime of Algorithm 2 in [15] is several orders of magnitude greater than the runtime of any of the other considered algorithms, and it also increases much faster in B. Due to its very large runtime we will only consider the sampling complexity of this algorithm and not its performance for noisy input data.

Next we examine the quality of the frequency and coefficient reconstructions for noisy input data. For Algorithm 1 and `fft` we assume that the we can only sample a 2π-periodic function $g = f + \eta$, where f has exact short support of length B, i.e., $c_\omega = 0$ if ω is not contained in the support, and $\eta\colon [0, 2\pi] \to \mathbb{C}$ satisfies that its vector of Fourier coefficients $\mathbf{c}(\eta) \in \mathbb{C}^N$ is uniformly distributed noise. For Algorithm 2 in [24] we create disturbed Fourier data $\widehat{\mathbf{y}} \in \mathbb{C}^N$ by adding uniform noise $\boldsymbol{\varepsilon} \in \mathbb{C}^N$ to $\widehat{\mathbf{x}}$,

$$\widehat{\mathbf{y}} := \widehat{\mathbf{x}} + \boldsymbol{\varepsilon}.$$

We measure the noise with the SNR value,

$$\text{SNR} := 20 \cdot \log_{10} \frac{\|\mathbf{c}(f)\|_2}{\|\mathbf{c}(\eta)\|_2} \quad \text{and} \quad \text{SNR} := 20 \cdot \log_{10} \frac{\|\widehat{\mathbf{x}}\|_2}{\|\boldsymbol{\varepsilon}\|_2}.$$

Recall that Algorithm 1 reconstructs the Fourier coefficients from function values, and that `fft` can be applied to the vector of N equidistantly chosen function values. Algorithm 2 in [24], on the other hand, recovers a vector from its Fourier transform, which means that their outputs are contained in different domains. Figures 2 and 3 depict the average reconstruction errors

$$\frac{\|\mathbf{x} - \mathbf{x}'\|_2}{N},$$

where $\mathbf{x}$ denotes the original Fourier transform or input vector, respectively, and $\mathbf{x}'$ the reconstruction by the corresponding algorithm applied to noisy input data for support lengths $B = 100$ and $B = 1000$. While the algorithm presented in this paper does not reconstruct the input function quite as well as Algorithm 2 in [24], whose reconstruction error is about half an order of magnitude smaller, it still returns results with slightly smaller reconstruction errors than `fft` for both support lengths $B = 100$ and $B = 1000$.

Another aspect we want to investigate are the sampling requirements. If obtaining samples of f takes a lot of computational effort, reducing the number of samples might be more important than reducing the runtime of the algorithm. Figure 4 shows the ratio between the number of used samples and the bandwidth $N = 2^{20}$ for varying support lengths for Algorithm 1, Algorithm 2 in [24], Algorithm 2 in [15] and `fft`.

One can see that Algorithm 1 is efficient from a sampling point of view for support lengths B that are up to one order of magnitude smaller than the bandwidth, as in that case less than the N samples necessary for an FFT are used. Algorithm 1 has a

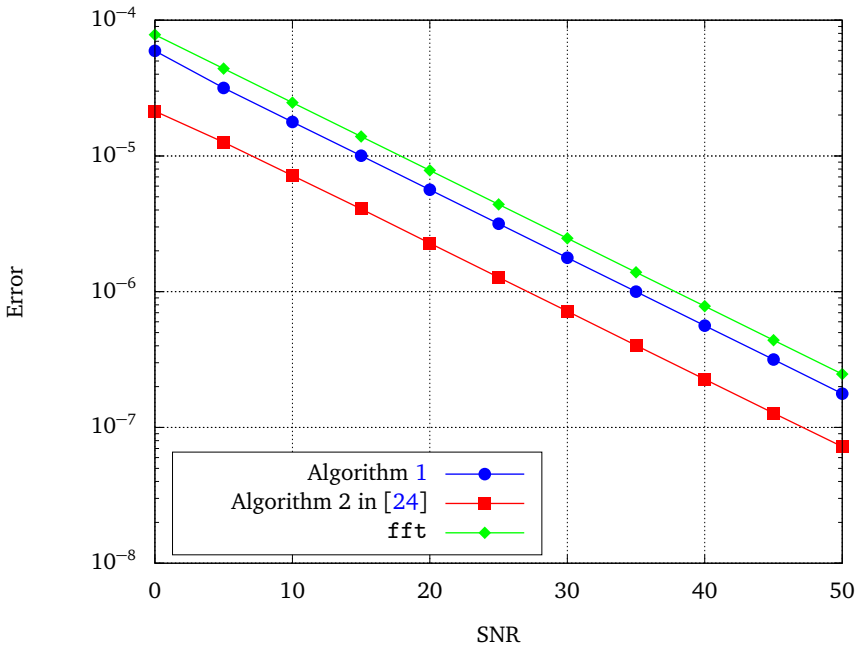

Figure 2: Average reconstruction errors $\|\mathbf{x}-\mathbf{x}'\|_2/N$ of Algorithm 1, Algorithm 2 in [24] and `fft` for 100 random input functions with uniformly distributed noise, support length $B = 100$, $N = 2^{20}$ and $\varepsilon = 10^{-4}$.

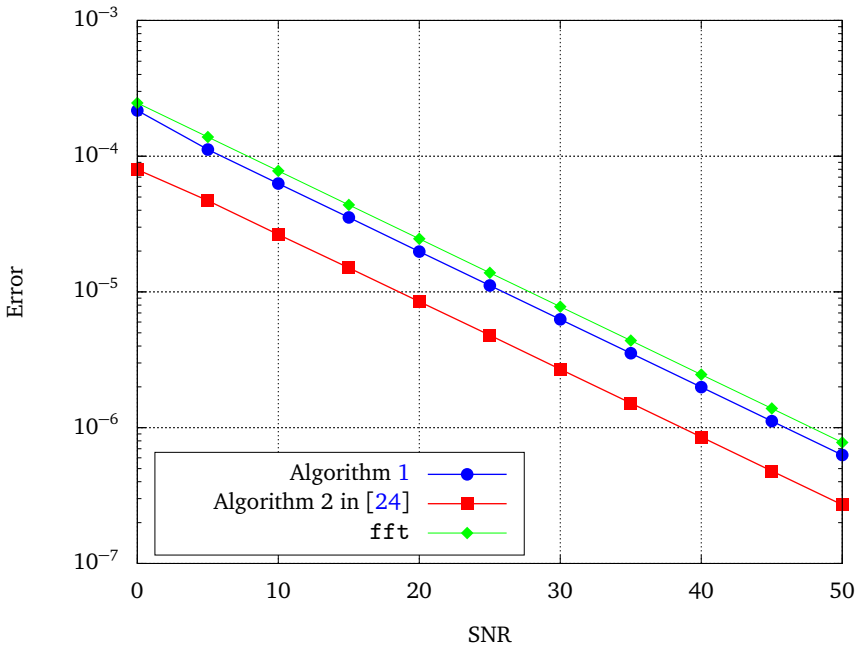

Figure 3: Average reconstruction errors $\|\mathbf{x}-\mathbf{x}'\|_2/N$ of Algorithm 1, Algorithm 2 in [24] and `fft` for 100 random input functions with uniformly distributed noise, support length $B = 1000$, $N = 2^{20}$ and $\varepsilon = 10^{-4}$.

samples per bandwidth ratio that is, even for $B = 100$, several orders of magnitude smaller than the one of Algorithm 2 in [15]. This is not surprising, since our method uses $L+1$ vectors that are essentially a subset of the $K(L+1)$ sampling vectors for the latter method, which already requires more than N samples for a function with sparsity 10 for $N = 2^{20}$.

If one just compares the number of used samples, the samples per bandwidth ratio of Algorithm 1 is even an order of magnitude greater than the one of Algorithm 2 in [24]. However, even though that algorithm only uses a small number of samples, it is apriori not known which entries of the input vector will be required, as they are found adaptively. Hence, Algorithm 2 in [24] has to be able to access the complete N-length Fourier data vector $\widehat{\mathbf{y}}$ as an input. Algorithm 1, on the other hand, uses

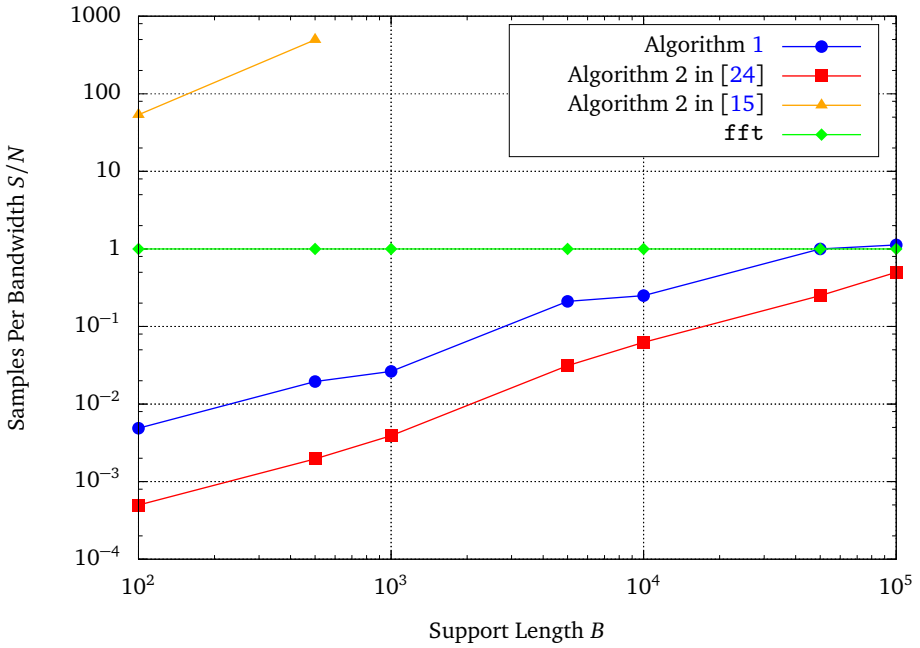

Figure 4: Number of used samples per bandwidth $N = 2^{20}$ for varying support lengths B for Algorithm 1, Algorithm 2 in [24], Algorithm 2 in [15] and `fft`.

all $\sum_{l=0}^{L} t_l s$ samples of f, but we can compute apriori from the bound B on the support length and the bandwidth N at which points of the form $\frac{2\pi j}{t_l s}$ the function f has to be evaluated.

5 Conclusion

In this paper we developed a deterministic fast discrete Fourier transform algorithm for the recovery of 2π-periodic functions with short frequency support. However, there are still some open questions. One could investigate whether related techniques can be used for other types of sparse input functions, e.g., functions where the energetic frequencies are contained in 2, 3 or n disjoint intervals of length B.

One of the reviewers suggested a different way of adapting Algorithm 2 in [15] for input functions with short support, namely to just use K separating primes $2B < s_1 < \cdots < s_K$ for which the CRT can be applied, instead of choosing one separating s and the small primes $t_1, \ldots, t_L$. Utilizing the short frequency support, it appears that this approach would have a runtime of

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

while using

$$\mathcal{O}\left(\frac{(B+\log N)\log N}{\log^2 B}\log\left(\frac{B+\log N}{\log B}\right)\right)$$

samples of the input function, which is even faster and requires fewer samples than the algorithm proposed herein. This lower sampling complexity could even lead to samples per bandwidth ratios that are smaller than the ones obtained for Algorithm 2 in [24]. We will investigate this method and possible implementations for noisy input data in the future.

Acknowledgment

The author gratefully acknowledges partial support for this work by the DFG in the framework of the GRK 2088 and thanks Gerlind Plonka for suggesting many helpful improvements to this paper. The author would also like to thank the anonymous reviewers for their careful reading and their valuable comments and suggestions that enhanced the previous version of the paper.

References

[1] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. *Proc. 23rd COLT*, pages 381-393, 2010.

[2] A. Akavia. Deterministic sparse Fourier approximation via approximating arithmetic progressions. *IEEE Trans. Inform. Theory*, 60(3):1733-1741, 2014.

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

[4] S. Bittens. SFFT for Functions with Short Frequency Support. http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software, 2017.

[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] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[7] E. Chu and A. George. *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. Computational Mathematics. Taylor & Francis, 1999.

[8] M. Frigo and S. G. Johnson. FFTW 3.3.6. http://www.fftw.org/, 2017.

[9] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan and M. Strauss. Near-optimal Sparse Fourier Representations via Sampling. *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 152-161, 2002.

[10] 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.

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

[12] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly Optimal Sparse Fourier Transform. *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 563-578, 2012.

[13] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and Practical Algorithm for Sparse Fourier Transform. *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1183-1194, 2012.

[14] S. Heider, S. Kunis, D. Potts and M. Veit. A sparse Prony FFT. *Proc. 10th International Conference on Sampling Theory and Applications (SAMPTA)*, pages 572-575, 2013.

[15] M. A. Iwen. Combinatorial Sublinear-Time Fourier Algorithms. *Found. Comput. Math.*, 10(3):303-338, 2010.

[16] M. A. Iwen. Improved Approximation Guarantees for Sublinear-Time Fourier Algorithms. *Applied and Computational Harmonic Analysis*, 34(1):57-82, 1 2013.

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

[18] S. Lang. *Algebraic Number Theory*. Graduate Texts in Mathematics. Springer New York, 2013.

[19] The MathWorks. MATLAB's documentation of `fft`. https://www.mathworks.com/help/matlab/ref/fft.html?searchHighlight=fft&s_tid=doc_srchtitle, 2017.

[20] Y. Mansour. Randomized Interpolation and Approximation of Sparse Polynomials. *ICALP*, 1992.

[21] J. Morgenstern. Note on a Lower Bound on the Linear Complexity of the Fast Fourier Transform. *J. ACM*, 20(2):305-306, 1973.

[22] S. Pawar and K. Ramchandran. Computing a k-sparse n-length discrete Fourier transform using at most $4k$ samples and $\mathcal{O}(k \log k)$ complexity. *IEEE International Symposium on Information Theory*, pages 464-468, 2013.

[23] G. Plonka and M. Tasche. Prony methods for recovery of structured functions. *GAMM-Mitt.*, 37(2):239-258, 2014.

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

[25] G. Plonka and K. Wannenwetsch. A sparse fast Fourier algorithm for real non-negative vectors. *Journal of Computational and Applied Mathematics*, http://doi.org/10.1016/j.cam.2017.03.019, 2017.

[26] 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.

[27] D. Potts, M. Tasche and T. Volkmer. Efficient Spectral Estimation by MUSIC and ESPRIT with Application to Sparse FFT. *Frontiers in Applied Mathematics and Statistics*, 2:1, 2016.

[28] K. Wannenwetsch and G. Plonka. Sparse FFT (small support). http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software, 2016.