A sparse multidimensional FFT for real positive vectors

Letourneau, Pierre-David† Langston, M. Harper Meister, Benoit
Lethin, Richard ‡
May 10, 2016

Abstract

We present a sparse multidimensional FFT randomized algorithm (sMFFT) for positive real vectors. The algorithm works in any fixed dimension, requires an (almost)-optimal number of samples \(O(R \log(N/R))\) and runs in \(O(R \log(R) \log(N/R))\) complexity (where \(N\) is the size of the vector and \(R\) the number of nonzeros). It is stable to noise and exhibits an exponentially small probability of failure.

1 Introduction

The Fast Fourier Transform (FFT) algorithm reduces the computational cost of computing the Discrete Fourier Transform (DFT) of a general complex \(N\)-vector from \(O(N^2)\) to \(O(N \log(N))\). Since its popularization in the 1960s [7], the fast Fourier Transform algorithm has played a crucial role in multiple areas of computational mathematics and has rendered possible great achievements in scientific computing [8], signal processing [24] and computer science [9] to name but a few. Such complexity has been shown to be optimal in the general case [26]. On the other hand, in more restricted cases, such as when the vector to be recovered is sparse, it is possible to significantly improve on the latter.

Indeed, the past decade or so has seen the design and study of various algorithms that can compute the DFT of sparse vectors using significantly less time and measurements than are traditionally required \([1, 3, 5, 10–22, 25, 27–29]\). That is, if \(f\) is a \(N \times 1\) vector corresponding to the DFT of noisy \(N \times 1\) vector \(\hat{f}\) containing at most \(R\) nonzero elements, it is possible to recover \(f\) using a number of samples much smaller than the traditional “Nyquist rate” (\(\ll O(N)\)), and in computational complexity much lower than that of the FFT (\(\ll O(N \log(N))\)).

These schemes are generally referred to as "sparse Fast Fourier Transform“ (sFFT) algorithms. They fall within two main categories: deterministic \([1, 3, 5, 10–22, 25, 27–29]\) or randomized \([2, 10, 12, 14, 18, 20, 26]\) algorithms. Of the two, randomized algorithms have had so far the most success in practice;

*This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Patent Pending. (U.S. Patent Application No. 62/286,732)

†Corresponding author: letourneau@reservoir.com

‡Reservoir Labs, 632 Broadway Suite 803, New York, NY 10012
although deterministic algorithms do exhibit polylogarithmic complexity in \( \log(N) \) and \( R \), i.e., a computational cost \( \leq C_{\text{det}} R \log^a \) for some positive algorithmic constant \( C_{\text{det}} > 0 \) and power \( a \geq 1 \), the algorithmic constant \( C_{\text{det}} \) is so big that they are only competitive when \( N \) is impractically large and when the sparsity satisfies stringent conditions. On the other hand, the algorithmic constant of randomized sparse FFTs, \( C_{\text{rand}} \), is in general much smaller, but depends on a parameter not present in a deterministic context: the probability of failure \( p \) of recovering the actual solution, i.e., \( C_{\text{rand}} \sim C_{\text{rand}}(p) \). As shown in Table 1 of all modern randomized algorithms, our algorithmic constant exhibits the most desirable behavior with respect to \( p \), i.e., \( C_{\text{sMFFT}} \sim \log(p) \).

Sparse FFT algorithms can further be split into a few more categories: 1D versus multidimensional \((d)\), and exact versus noisy measurements. In particular, this paper is interested solely in cases that are stable to noise (noisy measurements) as encountered in most applications. Tables 1 and 2 summarize the properties of the most recent and most efficient sparse FFT algorithms in regards to these characteristics, namely, the required number of samples, the computational complexity, the number of unknowns \( N \) (power of 2 or not), whether it is random or deterministic, the functional dependence of algorithmic constant \( C \) on ambient dimension \((d)\) and probability of failure \((p)\). As shown in Table 2, all scaling reported neglect any \( \log(\log(N)) \) (or slower) factor, and that the functional behavior of the algorithmic constant is only reported when it is available.

| Reference | Time | Samples | \( N \) | Randomization | \( d \) |
|----------|------|---------|-------|---------------|------|
| [18] | \( O(R \log(N)^2) \) | \( O(R \log(N)) \) | \( 2^c \) | Random, \( C \sim \frac{1}{p^2} \) | 1D |
| [16] | \( O(N \log(N)^2) \) | \( O(R \log(N)) \) | \( 2^c \) | Random | Any, \( C \sim d^d \) |
| [10] | \( O(R \log(N)^2) \) | \( O(R \log(N)) \) | \( 2^c \) | Random | 1D and 2D |
| [12] | \( O(R \text{poly}(\log(N))) \) | \( O(R \text{poly}(\log(N))) \) | \( 2^c \) | Random, \( C \sim \log(p^{-1}) \) | Any, \( C \sim 2^{O(d)} \) |
| [17] | \( O(R \log^{d+2}(N)) \) | \( O(R \log(N)) \) | \( 2^c \) | Random | Any, \( C \sim 2^{O(d)} \) |
| [17] | \( O(R \log(R) \log(\frac{N}{R})) \) | \( O(R \log(\frac{N}{R})) \) | \( 2^c \) | Deterministic | 1D |
| [17] | \( O(R \log(N)) \) | \( O(R) \) | \( 2^c \) | Deterministic | 1D |
| This paper | \( O(R \log(R) \log(\frac{N}{R})) \) | \( O(R \log(\frac{N}{R})) \) | Any | Random, \( C \sim \log(p) \) | Any, \( C \sim O(d) \) |

Table 1: Computational characteristics of recent sparse FFT algorithms

| Reference | Comment |
|----------|---------|
| [18] | Probability of failure \( p \leq \frac{1}{N^{O(\log(N))}} \) decays slowly with \( N \). |
| [16] | Average case only. |
| [10] | For accuracy \( \epsilon \), \( C \sim \epsilon^{-1} \). |
| [12] | Probability of failure \( p \leq \frac{1}{\log^2(N)} \) decays extremely slowly with \( N \). |
| [17] | it a priori knowledge of sparsity required. Support must contained within a cyclic interval. |
| [28] | Real positive vector. Support must contained within a cyclic interval. |

Table 2: Further comments on recent sparse FFT algorithms

The sMFFT algorithm we present in this paper has the following properties:

- Requires \( O(R \log(\frac{N}{R})) \) samples and \( O(R \log(R) \log(\frac{N}{R})) \) computational complexity (Section 3).
• Works in any dimension $d$ with cost linearly dependent on dimension (Section 4).
• Exponentially small probability of failure (Section 3).
• Stable to noise (Section 3.4).
• Works for any positive integer $N$ (Section 3).
• Works for positive real vectors only

In this sense, our algorithm matches the optimal sampling complexity [14] (within a $\log(\log(N))^{-1}$ factor), and the best time complexity achieved so far (see [28, 29]).

The algorithms of [28, 29] are however designed solely for the 1D case, and require that the support of the sparse vector be contained in some cyclic interval of size $O(R)$, an hypothesis that is extremely restrictive and that can only be met in very special situations. Therefore, although they possess the best time complexity in Table 1, their assumptions are so restrictive as to render them almost completely useless in practice. In opposition, the algorithm presented here does not make any assumption on the support of the sparse vector other than being sparse, and does achieve identical low complexities.

In addition to this, our algorithm is the first multidimensional version of a sparse FFT that scales linearly with dimension. Indeed, most previous endeavors were targeted at the 1D sparse FFT, whereas the only currently-existing multidimensional versions [12, 16, 17] have a cost that scales at least exponentially with dimension.

As for guarantees of recovering the correct solution (with probability $\geq 1 - p$), it is seen from Table 2 that the dependence of the algorithmic constant/probability of failure of randomized algorithms on the latter is often quite unfavorable. When estimates are available, the constant appears to behave like $\frac{1}{p^2}$ at best. Otherwise, $p$ is bounded above by an expression that decays very slowly as $N$ goes to infinity and might well prove unsatisfactory for practical problem sizes.

Finally, our algorithm does share with [29] the characteristic of being designed for real positive vectors only. This however is not such an impediment since in many applications of interest, e.g. radar imaging [23] (positive reflectivity), this is a valid hypothesis.

This document is structured as follows: in Section 2 we introduce the notation and a quantitative description of the problem. In Section 3 we describe the algorithm in the one-dimensional case. Section 3.1–3.3 provide more details and results about the crucial parts of the algorithm for the noiseless case. The case of noisy data is discussed in Section 3.4. Finally, we describe how to go from one dimension to multiple dimensions in Section 4. All proofs can be found in Appendix A.
2 Notation and description of the problem

In this section, we introduce the notation we shall use throughout the remainder of the paper. All quantities are summarized in Table 3.

| Symbol | Description |
|--------|-------------|
| \( f(\cdot) \) | Periodic, bandlimited function of interest. |
| \( \hat{f} \) | Vector containing Fourier coefficients of \( f(\cdot) \). |
| \( S \) | Set containing indices of nonzero elements of \( \hat{f} \), i.e., support |
| \( N \) | Total number of unknowns |
| \( R \) | Upper bound on cardinality of the support \( S \) of \( \hat{f} \) |
| \( d \) | Ambient dimension |
| \( p \) | Probability of failure of randomized algorithm |
| \( C \) | Algorithmic constant |

Table 3: Summary of important quantities

For all that follows, and unless otherwise stated, the function \( f(x) \) will be assumed to take the form,

\[
f(x) = \sum_{j=0}^{N-1} e^{-2\pi i x j} \hat{f}_j,
\]

for some finite \( 0 < N \in \mathbb{N} \). Furthermore, it is assumed that the vector \( \hat{f} \) has real and positive elements \( \{\hat{f}_j\}_{j=0}^{N-1} \geq 0 \) as well as positive sparsity level smaller than or equal to \( R < N \). That is, if we define the support of \( \hat{f} \) as,

\[
S := \left\{ n \in \{0,1,...,N-1\} : |\hat{f}_n| \neq 0 \right\},
\]

then,

\[
0 \leq \#S \leq R < N < \infty,
\]

and,

\[
\hat{f}_j \begin{cases} > 0 & \text{if } j \in S \\ = 0 & \text{o.w.} \end{cases}
\]

where \( \# \) indicates the cardinality of a set. In particular, we are interested in the case where \( R \ll N \). where \( \# \) indicates the cardinality of a set. In particular, we are interested in the case where \( R \ll N \). We shall denote by \( \mathcal{F} \) the inverse Fourier transform of an \( L^2(\mathbb{R}^d) \) function. That is,

\[
\mathcal{F} \left[ \hat{f}(\xi) \right] (x) = \int_{\mathbb{R}^d} e^{-2\pi i x \cdot \xi} \hat{f}(\xi) \, d\xi.
\]

The size-\( N \) Discrete Fourier Transform (DFT) is defined through,

\[
f_{n,N} = \left[ F_N \hat{f} \right]_n = \frac{1}{N} \sum_{j=0}^{N-1} e^{-2\pi i \frac{n}{N} j} \hat{f}_j, \quad n = 0, 1, ..., N - 1.
\]
With such notation in hand we can state the problem we are solving as follows: let \( \hat{f} \) be some vector satisfying Eq. (2), and assume samples take the form of a “clean” signal plus some additive noise,

\[
f_n = f_{n,N}^{\text{noiseless}} + \eta \nu_n = \sum_{j=0}^{N-1} e^{-2\pi i \frac{2j}{N}} \hat{f}_j + \eta \nu_n
\]

for some small parameter \( 0 < \eta < 1 \), and where \([\nu_0, \nu_1, ..., \nu_{N-1}]^T\) has 1-norm on the order of \( \mathcal{O}(1) \). Then, find some vector \( \tilde{f} \) such that,

\[
\frac{||\hat{f} - \tilde{f}||_2}{||\hat{f}||_2} \leq \mathcal{O}(\eta)
\]

from the knowledge of \( \mathcal{O}(R \log \left(\frac{N}{R}\right)) \) measurements, and in \( \mathcal{O}(R \log(R) \log \left(\frac{N}{R}\right)) \) computational steps. We shall refer to the case where \( \eta = 0 \) as the noiseless case. Otherwise, we will say that the data is noisy.

Finally, we shall assume that one has access to a decomposition of \( N \), the total size of the spectrum, of the form,

\[
N = K \prod_{i=1}^{P} \rho_i
\]

where \( 2 \leq \rho_i \leq \rho = \mathcal{O}(1) \) for \( i = 1, 2, ..., P \), \( K = \mathcal{O}(\rho R) \) and \( P = \mathcal{O}(\log \left(\frac{N}{K}\right)) \). If no such decomposition exists, it is always possible to slightly increase the value of \( N \) and “pad with zeros” so that the resulting integer possesses such decomposition. As we will see, since the dependence on \( N \) is only logarithmic, such increase has barely any impact on the computational cost.

### 3 A sparse FFT in 1D

In this section, we describe a fast way to compute the one-dimensional DFT of a bandlimited and periodic function \( f(x) \) satisfying Eq. (1)-(2). Our approach to this problem can be broken into two separate steps: in the first step (Section 3.1-3.2), we are merely interested in recovering the support \( S \) of the vector \( \hat{f} \), whereas in the second step (Section 3.3), we rapidly compute the nonzero values of \( \hat{f} \) using the knowledge of the support acquired in the first step. We describe the algorithm in the noiseless case in this section and follow with a discussion of its stability in the noisy case in Section 3.4. Pseudo-code is provided in Algorithm 1-4.

**Algorithm 1 1DSFFT\((R, N)\)**

1. \( S \leftarrow \text{FIND\_SUPPORT}(R, N) \)
2. \( \hat{f} \leftarrow \text{COMPUTE\_VALUES}(S) \)
3. Output: \( \hat{f}, S \).
3.1 Finding the support

For all that follows in this section, we shall refer to the example in Figure 1. From a high-level perspective, our support-finding scheme makes use of three major ingredients: 1) sub-sampling, 2) shuffling, and 3) low-pass filtering. Sub-sampling allows us to reduce the size of the problem to a manageable level. Unfortunately, sub-sampling inevitably leads to aliasing. In other words, although it is possible to rapidly compute the FFT of a sub-sampled signal, this inevitably leads to the recovery of an aliased version of the original signal. This however, is not totally useless. Indeed, when Eq. (2) is satisfied, an aliased Fourier coefficient is nonzero if and only if its corresponding aliased lattice contains an element of the true support (note that positivity is crucial here to avoid cancellation). This provides a useful criterion to discriminate between elements that belong to the support and elements that do not.

We show how this is useful through a concrete example, shown in Figure 1. Line 0 represents a lattice (represented by the thin tickmarks) of size,

\[ N = 40 = 5 \prod_{i=1}^{3} 2 = K \prod_{i=1}^{P} \rho_i \]

with \( K = 5 \), \( \rho_i = 2 \ \forall \ i \) and \( P = 3 \), which contains only 3 positive frequencies (represented by black dots in \( S = \{1, 23, 35\} \)). Then, consider line 1 which corresponds to the aliasing modulo 10 of the previous sets. In this case, knowledge of the nonzero coefficients (\( \{1, 3, 5\} \)) and the fact that the signal is aliased modulo 10 (with a maximum of \( N = 40 \)) tells us that the support of the original signal must belong to,

\[ \{1, 11, 21, 31\} \cup \{3, 13, 23, 33\} \cup \{5, 15, 25, 35\} \]

Although this is not sufficient information to characterize the support, this is a reduction in the number of possibilities and a step forward. This simple “de-aliasing” scheme can be leveraged to produce an FFT algorithm for sparse positive vectors. Unfortunately, although the resulting algorithm possesses better scaling than the traditional FFT, its computational characteristics are far from optimal.

To reach better scalings, we perform shuffling and low-pass filtering. From a high-level, shuffling will distribute the frequency content more or less uniformly (with nonzero aliased frequencies “far” from each other) and low-pass filtering will allow for inexpensively reaching the Fourier coefficients through a small FFT. In turn, the information gained from performing these steps allows us to compute the location of the nonzero aliased components. Once the locations of the aliased nonzero frequencies have been identified, we fall back onto the previous “de-aliasing” scheme. This is the rationale behind the method.

A pseudo-code version of the support-recovering algorithm can be found in Algorithm 2. We now proceed to a detailed quantitative description of the procedure. Before we do so however, let us introduce a few definitions: Let \( k, N, M_k \in \mathbb{N} \), \( 0 < \alpha < 1 \) and \( S_k, \mathcal{W}_k, \mathcal{M}_k \subset \{0, 1, \ldots, N - 1\} \). Then,

- The aliased support \( S_k \) at step \( k \) corresponds to the indices of the elements of the true support \( S \) modulo \( M_k \).
• The working support at step $k$ corresponds to the set $W_k := \{0, 1, ..., M_k - 1\}$.

• A candidate support $M_k$ at step $k$ is any set satisfying $S_k \subset M_k \subset W_k$ of size $O(\rho R)$.

At the start of the process (step $k = 0$) only the fact that $S \subset \{0, 1, ..., N - 1\}$ is known. This corresponds to line 0) in Figure 1 where the black dots represent the location elements of $S$. The first step ($k = 1$) is performed as follows: letting $M_1 = \frac{N}{\prod_{i=2}^{P_i} \rho_i} = \rho_1 K = O(R)$, sample $f(x)$ at $x_{n_1} \prod_{i=2}^{P_i} \rho_i; N = \frac{n_1 \prod_{i=2}^{P_i} \rho_i}{N} = \frac{n_1}{M_1} = x_{n_1; M_1},$

for $n_1 \in \{0, 1, ..., M_1 - 1\}$ to get,

$$f_{n_1} \prod_{i=2}^{P_i} \rho_i; N = \sum_{j=0}^{N-1} e^{-2\pi i \frac{n_1 \prod_{i=2}^{P_i} \rho_i j}{N}} \hat{\hat{f}}_j = \sum_{l=0}^{M_1-1} e^{-2\pi i \frac{n_1 \prod_{i=2}^{P_i} \rho_i l}{M_1}} \left( \sum_{j: j \mod M_1 = l} \hat{\hat{f}}_j \right) = \sum_{l=0}^{M_1-1} e^{-2\pi i \frac{n_1 \prod_{i=2}^{P_i} \rho_i l}{M_1}} \hat{\hat{f}}^{(1)}_j$$

for $n_1 \in M_1 := \{0, 1, ..., M_1 - 1\}$ defined as the candidate support in the first step. From the last two lines, it is seen that the samples correspond to a DFT of size $M_1$ of the vector $\hat{\hat{f}}^{(1)}$ with entries that are an aliased version of those of the original vector $\hat{\hat{f}}$, as described previously. These can be computed through the FFT in order $O(M_1 \log(M_1)) = O(R \log(R))$. In this first step, it is further possible to rapidly identify the aliased support $S_1$ from the knowledge of $\hat{\hat{f}}^{(1)}$ since the former correspond to the set,

$$\{l \in \{0, 1, ..., M_1 - 1\} : \hat{\hat{f}}^{(1)}_l \neq 0\}.$$

This is a consequence of the fact that $\hat{\hat{f}}^{(1)}_l := \sum_{j: j \mod M_1 = l} \hat{\hat{f}}_j > 0$

if and only if $l \in S_1$, itself a consequence of the fact that Eq.(2) holds. In our example,

$$M_1 = \rho_1 K = 2 \cdot 5 = 10$$

, which leads to

$$S_1 = \{1 \mod 10, 23 \mod 10, 35 \mod 10\} = \{1, 3, 5\} = \{l \in \{0, 1, ..., 9\} : \hat{\hat{f}}^{(1)}_l \neq 0\};$$

$$W_1 = M_1 = \{0, 1, ..., 9\}.$$
The elements of $S_1$ are represented by thick tickmarks on line 1) of Figure 1, whereas the elements of the candidate support $M_1$ are represented by thin tickmarks. For this first step, the working support $W_1$ is equal to the candidate support $M_1$.

\[
0 \leq n \leq 40
\]

\[
0 \leq m \leq 10
\]

\[
0 \leq l \leq 20
\]

\[
0 \leq n \leq 40
\]

Figure 1: Computing the support $S$. Line 0): Initialization; elements of $S$ correspond to black dots and lie in the grid \{0, 1, ..., $N - 1$\}. Line 1): First step; elements of the candidate support $M_1$ are represented by thin tickmarks and those of the aliased support $S_1$ by thick tickmarks. $S_1$ is a subset of $M_1$ and both lie in the working support \{0, 1, ..., $M_1 - 1$\}. Line 2): Second step; elements of the candidate support $M_2$ correspond to thin tickmarks and are obtained through de-aliasing of $S_1$. Elements to the aliased support $S_2$ correspond to thick tickmarks. Both lie in the working support \{0, 1, ..., $M_2 - 1$\}. $M_2$ is a constant factor of $M_1$. Line 3): The final step correspond to the step when the working is equal to \{0, 1, ..., $N - 1$\}. step 2. Last step because $N = \alpha M_2$.

At this point, we can proceed to the next step ($k = 2$) as follows: we let,

\[ M_2 = \rho_2 M_1 = K \prod_{i=1}^{2} \rho_i = 5 \cdot 2^2 = 20 \]

and consider the samples,

\[ f_{n_2} \prod_{\ell=0}^{\rho_1-1} \rho_i; N = \sum_{l=0}^{M_2-1} e^{-2\pi i \frac{n_2 l}{N}} \left( \sum_{j:j \mod M_2 = l} \hat{f}_j \right) = \sum_{l=0}^{M_2-1} e^{-2\pi i \frac{n_2 l}{M_2}} \hat{f}_l^{(2)} = f_{n_2; M_2} \]

for $n_2 = 0, 1, ..., M_2 - 1$ as before. Here however, knowledge of $S_1$ is incorporated. Indeed, since $M_2$ is a multiple of $M_1$, it follows upon close examination that,

\[ S_2 \subset \bigcup_{k=0}^{\rho_1-1} (S_1 + kM_1) := M_2. \]

That is, the set $M_2$, defined as the union of $\rho_1 = O(1)$ translated copies of $S_1$, must itself contain $S_2$. Furthermore, it is of size $O(\rho_1 \# S_1) = O(\rho R)$ by construction. It is thus a proper candidate
support (by definition). In our example, one gets,
\[ \bigcup_{k=0}^{1} (S_1 + k \cdot M_1) = \{1, 3, 5\} \cup \{1 + 10, 3 + 10, 5 + 10\} = \{1, 3, 5, 11, 13, 15\} = M_2, \]
which indeed contains the aliased support,
\[ S_2 = \{1 \mod 20, 23 \mod 20, 35 \mod 20\} = \{1, 3, 15\}. \]
The elements of the former are represented by thin tickmarks on line latter are represented by thick tickmarks. The working support now becomes \( W_2 := \{0, 1, ..., 19\} \).
Once again, it is possible to recover \( S_2 \) by leveraging the fact that,
\[ \{l \in \{0, 1, ..., M_2 - 1\} : \hat{f}_l^{(2)} \neq 0\} = S_2. \]
Here however, the cost is higher since computing \( \hat{f}^{(2)} \) involves performing an FFT of size,
\[ M_2 = K \prod_{i=1}^{P} \rho_i = 5 \cdot 2 = 10 \]
and it is not difficult to see that the cost will increase exponentially if we keep acting in this fashion. For this reason, additional steps are required to bring the cost of identifying \( S_2 \) from \( M_2 \) down. Such steps are described in Section 3.2, and involve a special kind of shuffling as well as a filtering of the samples followed by an FFT of size \( O(R) \). All in all, it is shown that it is possible to recover \( S_k \) from the knowledge of \( M_k \) at any step \( k \) using merely \( O(R) \) samples and \( O(R \log(R)) \) computations.

Following the rapid recovery of \( S_2 \), we proceed in a similar fashion \( (P = O(\log(\frac{N}{R}))) \) times until \( W_k := \{0, 1, ..., N-1\} \) at which point \( S_k = S \). Note in particular that the size of the aliased support \( S_k \) and candidate support \( M_k \) remain of order \( O(R) \) and \( O(\alpha R) \) respectively at each step while the size of the working support increases exponentially fast; i.e., \( \#W_k = O(K \prod_{i=1}^{k} \rho_i) \geq 2^k \cdot R. \)
Since going from step \( k \) to step \( k + 1 \) (computing \( S_k \) from \( M_k \) and de-aliasing) can be done using \( O(K) = O(R) \) measurements and \( O(R \log(R)) \) time (Section 3.2), this implies a cost of
\[ O\left( R \log\left( \frac{N}{R} \right) \right) \]
measurements and
\[ O\left( R \log(R) \log\left( \frac{N}{R} \right) \right) \]
time to identify the support \( S \). The steps of this support-recovery algorithm are described in Algorithm 2. The correctness of the algorithm is guaranteed by the following proposition.

**Proposition 1.** In the noiseless case, Algorithm 2 outputs \( S \), the support of the vector \( \hat{f} \) satisfying Eq. (2), using,
\[ O\left( \log(p) R \log\left( \frac{N}{R} \right) \right) \]
samples and,
\[ O\left( \log(p) R \log(R) \log\left( \frac{N}{R} \right) \right) \]
time with probability at least \( (1 - p) \).
Proof. Refer to Proposition 4 in Appendix A.1 Section 3.2 as well as previous discussion.

From the knowledge of $\mathcal{S}$, it is possible to recover the actual values of $\hat{f}$ rapidly and with few samples. This is the second major step of the sMFFT and is described in detail in Section 3.3.

Algorithm 2 FIND\_SUPPORT(R, N)

1: Pick $0 < \alpha < \frac{1}{2}$ and $0 < p, \delta \ll 1$.
2: Let $N = K \prod_{i=1}^{P} \rho_i$ large enough to contain the original domain.
3: Let $M_1 = K$ and $M_1 = W_1 := \{0, 1, \ldots, M_1 - 1\}$.
4: for $k$ from 1 to $P$ do
5: \quad $S_k \leftarrow$ FIND\_ALIASED\_SUPPORT($M_k, \alpha, p, \delta$)
6: \quad $M_{k+1} := \bigcup_{m=0}^{p_k-1} (S_k + mM_k)$.
7: end for
8: Output: $S_L$.

3.2 Rapid recovery of $S_k$ from knowledge of $M_k$

In this section, we describe how to solve the problem of rapidly recovering the aliased support $S_k$ from the knowledge of a candidate support $M_k$ at step $k$. This correspond to function FIND\_ALIASED\_SUPPORT(·) in Algorithm 2. Before providing details, a few definitions are introduced.

Definition 1. Given $j, K \in \mathbb{Z}$, $K \neq 0$, define,

$$\chi_K(j) = \left\lfloor \frac{j}{K} \right\rfloor$$

where $\lfloor \cdot \rfloor$ indicates rounding to the nearest integer.

Definition 2. Let $0 < M \in \mathbb{N}$. Then, the set $Q(M)$ is defined as,

$$Q(M) := \{q \in [0, M) \cap \mathbb{Z} : q \perp M\}$$

where the symbol $\perp$ between two integers indicates they are coprime.

Algorithm 3 shows how we solve the aliased support recovery (from the candidate support) problem rapidly. Proofs of the correctness of the algorithm is provided in Proposition 2 which relies on results found in Appendix A.1. First, we provide a brief description of the algorithm followed by a concrete example. The aliased support recovery algorithm contains four major steps:

1. Permute samples randomly.
2. Apply a diagonal Gaussian filter.
3. Compute a small FFT.
4. Identify aliased support.
To help the reader better understand what each step of the aliased support recovery algorithm does, we once again proceed through an example. To begin with, assume that $M_k = 40$ and,

\[ S_k = \{1, 23, 35\} \]
\[ M_k = \{1, 3, 15, 21, 23, 35\} \]
\[ W_k = \{1, 2, ..., 39\} \]

as in step $k = 3$ of the previous section.

The first step is to randomly shuffle the elements of $M_k$ within $W_k$. We do so by applying a permutation operator $\Pi_Q(\cdot)$ in sample space,

\[ \Pi_Q \left( f^{(k)}_{n; M_k} \right) = f^{(k)}_{(nQ) \mod M_k; M_k} \]

for some integer $Q \in \mathcal{Q}(M_k)$. By Lemma 3 (Appendix A.1), this is equivalent to shuffling in frequency space as,

\[ \hat{f}^{(k)}_l \rightarrow \hat{f}^{(k)}_{(l[Q]_{M_k}^{-1}) \mod M_k} \]

where $[Q]_{M_k}^{-1}$ is the unique integer belonging to $\{0, 1, ..., M_k - 1\}$ such that $[Q]_{M_k}^{-1}Q \mod M_k = 1$. Furthermore, Lemma 5 (Appendix A.1) shows that if $Q$ is chosen uniformly at random within $\mathcal{Q}(M_k)$, the mapped elements of the candidate support $M_k$ will end up being more or less uniformly distributed within the working support $W_k$, i.e.,

\[ P \left( \left| (i[Q]_{M_k}^{-1}) \mod M_k - (j[Q]_{M_k}^{-1}) \mod M_k \right| \leq C \mid i \neq j \right) \leq O \left( \frac{C}{M_k} \right). \]

Thus, shuffling the indices of the samples through $\Pi_Q(\cdot)$, where $Q$ is picked randomly, is synonymous with uniform shuffling of the indices in Fourier space.

In our example, assume $Q \in \mathcal{Q}(M_k)$ is such that $[Q]_{M_k}^{-1} = 13$. Then, the sets $S_k$ and $M_k$ are mapped to,

\[ S_k^{\text{shuffled}} = \{(1 \cdot 13) \mod 40, (23 \cdot 13) \mod 40, (35 \cdot 13) \mod 40\} = \{13, 15, 19\} \]
\[ M_k^{\text{shuffled}} = \{(1 \cdot 13) \mod 40, (3 \cdot 13) \mod 40, (15 \cdot 13) \mod 40, (21 \cdot 13) \mod 40, (23 \cdot 13) \mod 40, (35 \cdot 13) \mod 40\} = \{13, 15, 19, 33, 35, 39\} \]

This is depicted on line B) of Figure 2 with the same convention as before.
Figure 2: Finding the aliased support $S_k$ from knowledge of $M_k$ (line A). First, indices are shuffled in value space leading to a shuffling in frequency space (line B). A Gaussian filter is applied followed by a small FFT (line C) on a grid $G$. The points of $M_k$ for which the value of the result of the last step at their closest neighbor in $G$ is small are discarded leaving only the aliased support $S_k$.

This shuffling step is followed by the application of a diagonal Gaussian filtering operator $\Psi_\sigma(\cdot)$ having elements $\{\hat{g}_\sigma \left( \frac{n}{M_k} \right) \}_n$ in sample space (step 2),

$$\Psi_\sigma [f_{n;M_k}] (n) = \hat{g}_\sigma \left( \frac{n}{M_k} \right) f_{n;M_k}. \quad (6)$$

By the properties of the Fourier transform, this is equivalent to convolution with the Gaussian,

$$g(x) = e^{-\frac{|x|^2}{\sigma^2}}$$

in frequency space. That is,

$$[\Psi_\sigma (\Pi_Q (f_{n;M_k}))] (m) = F \left( \sum_{j \in S_k} \hat{f}_{j}^{(k)} e^{-\frac{|n-(jQ)^{-1}_{M_k \mod M_k}|^2}{\sigma^2}} \right) (m). \quad (7)$$

for $n \in \{0, 1, ..., M_k - 1\}$, where we recall that $F[\cdot](n)$ is the Fourier transform operator evaluated at $n$. This expression is subsequently discretized to produce a DFT of a permuted version of the original Fourier coefficients convoluted with a Gaussian, i.e.,

$$\Psi_\sigma (\Pi_Q (f_{n;M_k})) = F_{M_k} \left( \sum_{j \in S_k} \hat{f}_{j}^{(k)} e^{-\frac{|n-(jQ)^{-1}_{M_k \mod M_k}|^2}{\sigma^2}} \right) + O(\delta)$$

where $\delta$ is a small numerical error (upon choosing $\sigma$ appropriately as dictated by Proposition 4). The result is shown on line C of Figure 2.

Further note that by design (through our choice of $\sigma$, refer to Proposition 4), the Gaussian filter is such that $|\hat{g}_\sigma \left( \frac{n}{M_k} \right)| \ll 1 \forall |n \mod M_k| \geq O(K)$. This in turn implies that only the quantities,

$$\left\{ \phi_n^{(k)}(Q) \right\}_{|n \mod M_k| \leq K} = \left\{ \sum_{j \in S_k} \hat{f}_{j}^{(k)} e^{-\frac{|n-(jQ)^{-1}_{M_k \mod M_k}|^2}{\sigma^2}} \right\}_{|n \mod M_k| \leq K}$$
should be computed. This can be done rapidly and accurately through a size-$K$ FFT leading to,
\[
[F_K (\Psi (\Pi_Q (f_n; M_k))))]_n = \sum_{j \in S_k} f_j^{(k)} e^{-\frac{|n - \left( (j|Q|^{-1} \mod M_k) \right)|^2}{2}} + O(\delta) = \phi_n^{(k)}(Q) + O(\delta) \quad (8)
\]
for $|n \mod M_k| \leq O(K)$. This small FFT corresponds to step 3 of the aliased support recovery algorithm.

Finally, by Proposition 4 for $\{Q^{(l)}\}$, i.i.d. uniform in $Q(M)$, the probability,
\[
P\left( \phi^{(k)}_j (Q^{(l)}) > O(\delta) \right)
\]
that an element $j \in M_k$ falls within a region where the DFT of the shuffled/filtered function is larger than $O(\delta)$ is small if $j \in M_k \cap S^c_k$ and equal to 1 if $j \in M_k \cap S_k$. This fact allows us to construct an efficient statistical test based on the knowledge of the quantities found in Eq. (3.2) to discriminate between the points of $M_k \cap S^c_k$ and those of $M_k \cap S_k$ (step 4 of the aliased support recovering algorithm). Such a test constitutes the core of Algorithm 3 and its correctness follows from the following proposition (proved in Appendix A.1),

**Proposition 2.** In the noiseless case, Algorithm 3 outputs $S_k$, the aliased support of the vector $\hat{f}$ at step $k$ with probability at least $(1 - p)$.

As for the cost, the permutation and multiplications step both incur a cost of $O(R)$. These are followed by an FFT of size $O(K) = O(R)$ which carries a cost of order $O(R \log(R))$. Finally, step 4 involves checking a simple property on each of the $O(\rho K)$ elements of $M_k$ incurring a cost $O(R)$. All of this is repeated $O(\log(p))$ times for a probability $(1 - p)$ of success. In conclusion, this implies that extracting $S_k$ from $M_k$ requires merely
\[
O(\log(p) R) = O(R)
\]
samples and,
\[
O(\log(p) R \log(R)) = O(R \log(R))
\]
computational time for fixed $p$, as claimed.
Algorithm 3 FIND_ALIASED_SUPPORT($\mathcal{M}_k, \alpha, p$)

1: Let $\mu$, $\Delta$ and $\eta$ be estimates for $\min_j \hat{f}_j$, $\frac{||f||_1}{\min_j f_j}$ and the noise level respectively.
2: Let $\frac{2n}{\mu} < \delta < 1$.
3: Let $\sigma = \frac{\alpha}{\sqrt{\log(\frac{2\Delta}{\delta})}}$, $K = O\left(\sigma \sqrt{\log(\frac{2\Delta}{\delta})}\right) \geq R$, and $L = \log_\alpha(p)$.
4: Let $S_k \leftarrow \mathcal{M}_k$
5: for $l$ from 1 to $L$ do
6: Compute: $\phi_{n}^{(k)}(Q(l))$, $n = 0, 1, ..., K - 1$.
7: for $j \in \mathcal{M}_k$ do
8: if $\left| \phi_{\mathcal{M}_k}^{(k)}(j)(Q(l)) \right| < \frac{\delta \mu}{2}$ then
9: Remove $j$ from $S_k$.
10: end if
11: end for
12: end for
13: Output: $S_k$.

### 3.3 Recovering values from knowledge of the support

In this section, we assume a set $S$ of size $\mathcal{O}(R)$ containing the support $S \subset \{0, 1, 2, ..., N - 1\}$ has been recovered. We discuss how to compute the values of the nonzero Fourier coefficients of $\hat{f}$ in Eq. (1) rapidly using this information. For this purpose, we assume that the Fourier transform of the function of interest can be sampled at locations

$$\left\{ \frac{qQ(t) \mod P(t)}{P(t)} \right\}_{p=1}^{P(t)}$$

for $t = 0, 1, ..., T$, $\{P(t)\}$ some prime numbers larger than or equal to $R$ and on the order of $\mathcal{O}(R)$ (their exact nature is discussed below), and $Q(t) \in \mathbb{Q}(P(t))$ for each $t$. This provides us with the following information,

$$\hat{f}_{qQ(t) \mod P(t)}; P(t) = \sum_{j \in S} e^{-2\pi i \frac{\hat{f}_j Q(t)}{P(t)}} \hat{f}_j [Q(t) \mod P(t)] = \sum_{l=0}^{P(t)-1} e^{-2\pi i \frac{q}{P(t)}} \left( \sum_{j \in S; j \mod P(t) = l} \hat{f}_j \right),$$

for $t = 0, 1, ..., T$. The outer sum is seen to be a DFT of size $P(t)$ of a shuffled and aliased vector, whereas the inner sum can be expressed as the application of a binary matrix $B_{q,j}^{(t)}$ with entries

$$B_{q,j}^{(t)} = \begin{cases} 1 & \text{if } j \mod P(t) = q \\ 0 & \text{o.w.} \end{cases}$$

to the vector $\hat{f}$ with entries $\{\hat{f}_j\}_{j \in S}$ corresponding to the nonzero frequencies of $\hat{f}$. In particular, each such matrix is sparse with exactly $\#S = \mathcal{O}(R)$ nonzero entries. Eq. (9) can be written in
matrix form as,

\[
(FB) \hat{f} = \begin{bmatrix}
F^{(1)} & 0 & \ldots & 0 \\
0 & F^{(2)} & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots \\
0 & 0 & \ldots & F^{(T)}
\end{bmatrix}
\begin{bmatrix}
B^{(1)} \\
B^{(2)} \\
\vdots \\
B^{(T)}
\end{bmatrix} \hat{f} = \begin{bmatrix}
f^{(1)} \\
f^{(2)} \\
\vdots \\
f^{(T)}
\end{bmatrix} = f^\dagger, \quad (10)
\]

where \(F^{(t)}\) is a standard DFT matrix of size \(P^{(t)}\). An important question that arises in this case is: does there exist a choice of numbers \(\{P^{(t)}\}\) such that \(FB\) is invertible and well-conditioned? The answer is yes as demonstrated by Theorem 1 (which relies on Proposition 5 and Corollary 2 found in Appendix A.1). What this result states is that if \(T \geq O(\log_R(N))\), then with nonzero probability,

\[(FB)^*(FB) = B^*B = I + \mathcal{P}\]

where \(I\) is the identity and \(\mathcal{P}\) is a perturbation with 2-norm smaller than \(\frac{1}{2}\). When this occurs, one can approached the linear system through the normal equation,

\[\hat{f} = \sum_{n=0}^{\infty} \mathcal{P}^n f^\dagger\]

which can be solved through the Neumann series,

\[\hat{f} = \sum_{n=0}^{\infty} \mathcal{P}^n f^\dagger.\]

This is exactly what Algorithm 4 does, and we have the following,

**Theorem 1.** Assume the support \(S\) of \(\hat{f}\) is known. Then Algorithm 4 terminates after \(O(\lceil \log(p) \rceil)\) iterations with probability greater than or equal to \(1 - p\).

**Proof.** By Proposition 5 (Appendix A.1), \(\|\mathcal{P}\| < \frac{1}{2}\) occurs with probability \(\geq \frac{1}{2}\). Thus, if we consider \(O(\log_2(p))\) independent realizations of \(FB\), the probability that at least one of them is invertible is greater than or equal to \(1 - p\). This is the reason for the outer while loop in Algorithm 4.

When this occur, Corollary 2 (Appendix A.1) states that the solution is given by the Neumann series. Furthermore,

\[
\left\| \hat{f} - \sum_{n=0}^{\lceil \log_2(\epsilon) \rceil} \mathcal{P}^n f^\dagger \right\| = \left\| \sum_{n=\lceil \log_2(\epsilon^{-1}) \rceil}^{\infty} \mathcal{P}^n f^\dagger \right\| \\
\leq \sum_{n=\lceil \log_2(\epsilon^{-1}) \rceil}^{\infty} \|\mathcal{P}\|^n f^\dagger \leq O(\epsilon)
\]

by the geometric series and the bound \(\|\mathcal{P}\| \leq \frac{1}{2}\). This underpins the choice of \(Z = \lceil \log(\epsilon) \rceil\) in Algorithm 4.

\(\square\)
As for the cost, since each matrix \( B^{(t)} \) contains exactly \( R \) nonzero entries, both \( B \) and \( B^*B \) can be applied in order \( RT = \mathcal{O}(R \log_R(N)) \) time. This is done \( \mathcal{O}(\log(\epsilon^{-1})) \) times for each realization. In addition, since \( F \) is a block diagonal matrix with \( T = \mathcal{O}(\log_R(N)) \) blocks consisting of DFT matrices of size \( \mathcal{O}(R) \), it can be applied in order \( \mathcal{O}(R \log(R) \log_R(N)) \) thanks to the FFT, and this must be done only once per realization. Therefore, the cost of computing the nonzero values of \( \hat{f} \) is bounded by,

\[
\mathcal{O} \left( \log(p) R \log_R(N) \left( \log(R) + \log(\epsilon) \right) \right),
\]

using at most,

\[
\mathcal{O} \left( \log(p) R \log_R(N) \right),
\]
as claimed.

**Algorithm 4** \( \text{COMPUTE\_VALUES}(S, R, \epsilon) \)

1. Let \( T = \mathcal{O}(4 \ceil{\log_R(N)}) \) and \( Z = \mathcal{O}(\ceil{\log_2(\epsilon)}) \).
2. \( \text{while} \) true \( \text{do} \)
3. \( \text{Pick } \{ P^{(t)} \}_{t=1}^{T} \) i.i.d. uniform r.v. chosen among the set containing the smallest \( 2R \) prime numbers greater than \( R \).
4. \( \text{Pick } \{ Q^{(t)} \}_{t=1}^{T} \) i.i.d. r.v. uniformly distributed in \( Q(P^{(t)}) \) respectively.
5. \( \text{Sample } \{ f_n; P^{(t)} \}_{n=0}^{p^{(t)}-1}, t = 1, ..., T \)
6. \( \text{Compute } \tilde{f}_0 \leftarrow (FB)^*f \)
7. \( \text{if } \| (B^*B) \tilde{f}_0 \|_2 < 1 \text{ then} \)
8. \( \tilde{f} \leftarrow \sum_{n=0}^{Z}(I - B^*B)^n \tilde{f}_0 \)
9. \( \text{Output: } \tilde{f} \).
10. \( \text{Exit.} \)
11. \( \text{end if} \)
12. \( \text{end while} \)

### 3.4 Stability

As discussed previously, the algorithms introduced in Section 3 have been designed for vectors which are \textit{exactly sparse}. In this section, we discuss the effect of noise and approximate sparsity. In fact, we show that when the noise is small enough, our algorithm recovers the support and the values of the nonzero frequencies of \( f(\cdot) \) with the same guarantees as described earlier.

#### 3.4.1 Support

The most important quantity for the fast recovery of the aliased support from a candidate support is Eq. (10), so let us re-visit it in the presence of noise. That is, consider

\[
\phi^{(k)}_n(Q) = [F^*_K (\Psi_{\sigma} (\Pi_Q ((f_n; M_k) + \eta v_n; M_k))))]_n. \tag{11}
\]

In particular, the error term can be \textit{uniformly bounded} following the next lemma,

**Lemma 1.** \textit{In the noisy case and under the assumptions of Section 2, the error term of the computed value } \( \phi^{(k)}_n(Q) \text{ takes the form,} \)

\[
\eta [F^*_K (\Psi_{\sigma} (\Pi_Q (v_n; M_k))))]_n
\]
and is uniformly bounded by an expression on the order of,

\[ \| \eta \left[ F_K \left( \Psi_\sigma \left( \Pi_Q \left( \nu_n, M_k \right) \right) \right] \|_\infty = O(\eta) \]

Now, recall that the crux of the statistical test in Algorithm 2 relies on the fact that if,

\[ |\phi_n^{(k)}(Q^{(i)})| > \delta \min_{j \in S} \hat{f}_j = \delta \mu \]

for multiple random \( Q^{(i)} \in Q(M) \), then the probability that \( n \) is an element of the (aliased) support is very high (and very low otherwise). This is quantified by Proposition 4 in the absence of noise. The presence of noise can skew this test in two ways: 1) by bringing the value of \( |\phi_n^{(k)}(Q^{(i)})| \) below the threshold when \( n \in S_k \) or 2) by bringing the value above the threshold multiple times when \( n \notin S_k \). Either way however, if it is possible to choose \( \delta \) such that,

\[ \eta < \frac{\delta \mu}{2} \]

then Lemma 1 guarantees that the noise can only affect the value so much, and the proof of Proposition 4 follows through with similar estimate after replacing \( \delta \mu \) by,

\[ \frac{\delta \mu}{2} \]

This is the reason for the choices of parameters in Algorithm 3.

Note also that this analysis states that it is not possible to recover elements of \( j \in S \) for which the value \( \hat{f}_j \) falls below the noise level, in agreement with one’s intuition. In this sense, in the presence of noise it might be more judicious to use a different notion of sparsity than the one presented in Section 2, namely one of approximate sparsity. In this case, \( R \) corresponds to the cardinality of the largest set \( S \) such that there exists some real number \( \eta > 0 \) such that

\[ \min_{j \in \hat{S}} \hat{f}_j \geq \eta \]
\[ \| \hat{f} - \hat{f}_\hat{S} \|_1 < \eta \]

where \( \hat{f}_\hat{S} \) is the restriction of \( \hat{f} \) to the set \( \hat{S} \), whenever such set exists.

3.4.2 Recovering values from knowledge of the support

It is quickly observed that the recovery of the values is a well-conditioned problem (with high probability). Indeed, since

\[ \frac{1}{T}(FB)^*(FB) = I - P, \]

and \( \|P\|_2 \leq \frac{1}{2} \) with high probability by Proposition 5 a simple argument based on the singular value decomposition produces the following corollary,

**Corollary 1.** Under the hypothesis of Proposition 5 \( \left( \frac{1}{T}(FB)^*(FB) \right)^{-1} \) exists, and

\[ \text{cond} \left( I - \frac{1}{T}(FB)^*(FB) \right) \leq 2 \]

in the \( \ell_2 \)-norm, with probability greater than or equal to \( 1 - p \).
Therefore, for noisy samples of the form,
\[ f_p(t) = f_p(t) + \eta \nu_p(t), \]
as described in Section 2, the output of Algorithm 4 is such that,
\[ ||\tilde{f} - \hat{f}||_2 \leq \eta \left\| \left( \frac{1}{T}(FB)\ast(FB) \right)^{-1} \right\|_\infty ||\nu||_1 = \mathcal{O}(\eta), \]
by Hölder inequality, since \( ||\nu||_1 = \mathcal{O}(1) \) by assumption and where \( \hat{f} \) is the exact solution. This proves stability.

4 The multi-dimensional sparse FFT

By nature, whenever dealing with the multidimensional DFT/FFT, it must be assumed that the function of interest is both periodic and bandlimited with fundamental period \([0, 1)\) in each dimension, i.e., that it has an expansion of the form
\[ f(x) = \sum_{j \in \{[0, M) \cap \mathbb{Z}\}^d} d^{-\frac{2\pi i x \cdot j}{M}} \hat{f}_j \]  
for some finite \( M \in \mathbb{N} \) and \( j \in \mathbb{Z}^d \). Computing its Fourier coefficients is then equivalent to computing the \( d \)-dimensional integral
\[ \hat{f}_n = \int_{[0,1]^d} e^{-2\pi i n \cdot x} f(x) \, dx. \]
This is traditionally achieved through a tensor product of quadratures, i.e., a trapezoidal quadrature rule in each dimension,
\[ \hat{f}_{(j_1, j_2, ..., j_d)} = \sum_{a=1}^{d} \sum_{n_a=0}^{M-1} \left( \prod_{k=1}^{d} e^{2\pi i \frac{kn_a}{M}} \right) f_{(n_1, n_2, ..., n_d)}, \]
which can be written as
\[ \hat{f}_{(j_1, j_2, ..., j_d)} = M^{-1} \sum_{n_1=0}^{M-1} e^{2\pi i \frac{j_1 n_1}{M}} \left( ... \left( \sum_{n_{d-1}=0}^{M-1} e^{2\pi i \frac{j_{d-1} n_{d-1}}{M}} \left( \sum_{n_d=0}^{M-1} e^{2\pi i \frac{j_d n_d}{M}} f_{(n_1, n_2, ..., n_d)} \right) \right) \right). \]
This can be recognized as a “dimension-by-dimension” DFT and is generally the formulation of choice for computing the multidimensional FFT [6,31].

However, this is not the only possibility. Indeed, through Proposition 3 below we show that it is possible to re-write the \( d \)-dimensional DFT as that of a 1-dimensional function with Fourier coefficients equal to those of the original function, but expressed in a different order.
Proposition 3. Assume the function $f : [0, 1)^d \rightarrow \mathbb{C}$ is bandlimited and periodic and has form 

\[ f(x) = \sum_{n=0}^{N-1} c_n e^{2\pi i j \cdot n x} \]

for all $j \in \{0, 1, ..., M-1\}$, where,

\[ x_n = \frac{ng \mod N}{N} \]

and $g = (1, M, M^2, ..., M^{d-1})$ and $N = M^d$.

This result has an interesting geometric interpretation, which we now describe. Eq. (13) shows that one can write the summation in two different ways due to periodicity, namely,

\[ \frac{1}{N} \sum_{n=0}^{N-1} e^{-2\pi i \frac{ng \mod N}{N}} f \left( \frac{ng \mod N}{N} \right) = \frac{1}{N} \sum_{n} e^{-2\pi i \frac{ng}{N}} f \left( \frac{ng}{N} \right) \]

The left-hand side represents a quadrature rule with points $x_n = \frac{ng \mod N}{N}$ distributed (more-or-less uniformly) in $[0, 1)^d$ as depicted on the left of Figure 3 (grey dots). The right-hand side represents an equivalent quadrature where the points $\tilde{x}_n = \frac{ng}{N}$ now lie on a line embedded in $\mathbb{R}^d$ as shown in the right side of Figure 3.

In each case, the bold, black squares represent the fundamental region of the periodic function $f(x)$, which can be repeated ad infinitum over the whole plane so as to tile its entirety without overlapping. The location at which the lattice (thin black lines) intersect represent the standard (tensor product) multidimensional DFT samples. Depending on the point of view one adopts, our discretization either corresponds to a uniform quadrature along the line $sg$, $s \in [0, 1)$ (right, Figure 3), or to a 2D quadratures with points corresponding to those found in the previous case folded back into the fundamental domain $[0, 1]^2$ (left, Figure 3). Note in particular that they are quite different from those obtained through the tensor product.

Proposition 3 therefore allows us to write any $d$-dimensional DFT as a one-dimensional DFT. It suffices to pick the appropriate sample points (Eq. (14)) and proceed to a re-ordering of the Fourier coefficients through the isomorphism

\[ \tilde{n} : \{ n \in \{0, M \} \cap \mathbb{Z}^D \} \rightarrow n_0 + n_1 M + ... + n_{D-1} M^{D-1} \in [0, M^D) \cap \mathbb{Z}^D. \]

5 Numerical results

We have implemented a prototype of our algorithm in MATLAB. In this section, we present a few numerical results which exhibit the claimed scaling. All simulation were carried out on a cluster possessing 4 Intel Xeon E7-4860 v2 processors and 256GB of RAM, with the MATLAB flag $\text{−singleCompThread}$ to ensure fairness through the use of a single computational thread.

The numerical experiments presented here fall in two categories:

\[^3\text{MATLAB} \text{ is a trademark of Mathworks}\]
1. Dependence of running time \((t)\) in seconds (s) as a function of the total number of unknowns \(N\) for a fixed number of nonzero frequencies \(R\).

2. Dependence of running time \((t)\) in seconds (s) as a function of the number of nonzero frequencies \(R\) for a fixed total number of unknowns \(N\).

All experiments were carried out in three dimensions (3D). The nonzero values of \(\hat{f}\) were picked randomly and uniformly in \([0.5, 1.5]\). The value of the parameters needed by Algorithm 1-4 were set according to Table 4. All running times are compared with those of the MATLAB `fftn` function, which uses a dimension-wise decomposition of the DFT (see Section 4) together with a 1D FFT routine along each dimension. Finally, note that no software optimization (such as cache optimization, vectorization, compilation and/or parallelization) were attempted.
Table 4: Values of parameters required by Algorithm 1-4 and used for numerical experiments

| Parameter | Description               | Value (Case 1) | Value (Case 2) |
|-----------|---------------------------|----------------|----------------|
| N         | Total number of unknowns  | variable       | 10^8           |
| R         | Number of nonzero frequencies | 50            | variable       |
| α         | Gaussian filter parameter | 0.15           | 0.15           |
| δ         | Statistical test parameter | 0.1           | 0.1            |
| p         | Probability of failure    | 10^{-4}        | 10^{-4}        |
| d         | Ambient dimension         | 3              | 3              |
| η         | Noise level               | 10^{-1}        | 10^{-1}        |

To study the dependence on the total number of points in the frequency lattice \((N, \text{case } 1)\), we picked \(R = 50\) nonzero frequencies distributed uniformly at random on a 3D lattice having \(N^{1/3}\) elements in each dimension, for different values of \(N \in [10^3, 10^{10}]\). Then, we simulated samples; for the MATLAB function, we computed the DFT on the full data cube, whereas for the sMFFT we simulated the samples at \(O(R \log(N))\) appropriate points dictated by the theory (Section 3). Finally, we ran both algorithms on the simulated data and recorded the running time and relative error of the solution (in the case of the sMFFT only; the FFT is exact up to machine accuracy).

The results are shown in Table 5 and Figure 4. As can be observed, the cost of computing the DFT through the sMFFT remains more or less constant with \(N\), whereas that the the MATLAB fftn() function increases linearly on a log-log plot. This is the expected behavior and demonstrates how advantageous it can be to use the sMFFT when the proper hypotheses are met, and especially in high dimensions. Also note that the maximum pointwise error observed among all experiments was \(9.3 \cdot 10^{-3}\) which is as expected given the noise level \((\eta = 10^{-1})\).

Table 5: sMFFT and MATLAB fftn() function running time dependence on number of unknowns \((N)\)

| log \((N)\) | Relative error - sMFFT | Running time - sMFFT (s) | Running time - fftn() (s) |
|------------|-------------------------|--------------------------|--------------------------|
| 3.6124     | 7.4 \cdot 10^{-3}      | 1.21 \cdot 10^{-2}      | 6.0 \cdot 10^{-4}       |
| 4.5154     | 9.3 \cdot 10^{-3}      | 1.04 \cdot 10^{-2}      | 1.5 \cdot 10^{-3}       |
| 5.4185     | 8.4 \cdot 10^{-3}      | 6.4 \cdot 10^{-3}       | 1.68 \cdot 10^{-2}      |
| 6.3216     | 7.3 \cdot 10^{-3}      | 7.9 \cdot 10^{-3}       | 1.32 \cdot 10^{-1}      |
| 7.2247     | 6.3 \cdot 10^{-3}      | 8.9 \cdot 10^{-3}       | 1.78                     |
| 8.1278     | 6.8 \cdot 10^{-3}      | 1.27 \cdot 10^{-2}      | 2.01 \cdot 10^{-1}      |
| 9.0309     | 6.1 \cdot 10^{-3}      | 1.30 \cdot 10^{-2}      | 1.68 \cdot 10^{-2}      |
Figure 4: Running time dependence on number of unknowns \((N)\) for the MATLAB \texttt{fftn} (black) and the sMFFT (red) in three dimensions (3D) and for \(R = 50\).

Our second experiment pertains to the behavior of the sMFFT for a fixed number of unknowns \((N)\) and a variable number of nonzero frequencies \((R, \text{Case 2})\). In this case, we fixed \(N = \mathcal{O}(10^8)\) and proceeded to compare the sMFFT algorithm with the MATLAB \texttt{fftn}(\cdot) function as before, with the fixed parameters displayed in Table 4.

The results are shown in Table 6 and Figure 5. In this case, the theory states that the sMFFT algorithm should scale quasi-linearly with the number of nonzero frequencies \(R\). A close look at Figure 5 shows that it is indeed the case; whereas \(R\) spans more than 3 order of magnitude, the running time barely spans 3. It should also be noted that the timing exhibits a lot of “jitter,” seemingly taking less time for larger number of nonzero frequencies at times. Recall however that the algorithm is randomized, and that these results pertains to a single realization. For instance, Algorithm 4 is guaranteed to converge after \(\mathcal{O}({\log}(p))\) iterations (with high probability). However, on some occasions it might stop after a mere iteration leading to a seemingly lower running time.

also, once again we note that the maximum pointwise error observed among all experiments was \(1.55 \cdot 10^{-2}\), which is within the limits guaranteed by the theory. Finally, the cost \texttt{fftn}(\cdot) function remains constant as it scales like \(\mathcal{O}(N \log(N))\) and is oblivious to \(R\).
Table 6: sMFFT and MATLAB \texttt{fftn(\cdot)} function running time dependence on number of nonzeros ($R$) for $N = 10^7$

| $R$  | Relative error - sMFFT | Running time - sMFFT (s) | Running time - \texttt{fftn(\cdot)} (s) |
|------|------------------------|--------------------------|----------------------------------------|
| 4    | $1.55e^{-2}$           | $5.0e^{-3}$              | 2.48                                   |
| 8    | $1.43e^{-2}$           | $6.3e^{-3}$              | 2.48                                   |
| 16   | $1.47e^{-2}$           | $8.0e^{-3}$              | 2.48                                   |
| 32   | $6.8e^{-3}$            | $1.04e^{-2}$             | 2.48                                   |
| 64   | $6.9e^{-3}$            | $9.5e^{-3}$              | 2.48                                   |
| 128  | $4.5e^{-3}$            | $3.06e^{-2}$             | 2.48                                   |
| 256  | $3.2e^{-3}$            | $3.51e^{-2}$             | 2.48                                   |
| 512  | $2.5e^{-3}$            | $6.18e^{-2}$             | 2.48                                   |
| 1024 | $1.8e^{-3}$            | $2.31e^{-1}$             | 2.48                                   |
| 2048 | $1.3e^{-3}$            | $2.71e^{-1}$             | 2.48                                   |
| 4096 | $0.9e^{-3}$            | 3.86                     | 2.48                                   |
| 8192 | $0.7e^{-3}$            | 3.65                     | 2.48                                   |
| 16384| $0.5e^{-3}$            | 2.83                     | 2.48                                   |

Figure 5: Running time dependence on number of nonzero frequencies ($R$) for the MATLAB \texttt{fftn(\cdot)} (black) and the sMFFT (red) in three dimensions (3D) and for $N = 10^8$. 

23
A Proofs

In this appendix, we present all proofs and accompanying results related to the statements presented in the main body of the work.

A.1 Proofs of Section 3.1 and Section 3.2

Lemma 2. Let \( 0 < Q \leq N \in \mathbb{N} \) with \( Q \perp N \). Then the map,

\[
    n \in \{0, 1, ..., N - 1\} \rightarrow nQ \mod N \subset \{0, 1, ..., N - 1\}
\]

is an isomorphism.

Proof. Since the range is discrete and a subset of the domain, it suffices to show that the map is injective. Surjectivity will then follow from the pigeon hole principle. To show injectivity, consider \( i, j \in \{0, 1, ..., N - 1\} \), and assume,

\[
    iQ \mod N = jQ \mod N
\]

This implies (by definition) that there exists some integer \( p \) such that,

\[
    (i - j)Q = pN
\]

so that \( N \) divides \((i - j)Q\). However, \( N \perp Q \) so \( N \) must be a factor \((i - j)\). However, since \( i, j \) are restricted to \( \{0, 1, ..., N - 1\} \), it must be that,

\[
    |i - j| < N
\]

and the only possibility is,

\[
    i - j = 0 \iff i = j
\]

This demonstrates injectivity and the result follows.

Lemma 3. Let \( 0 < Q < M \) be an integer coprime to \( M \) and,

\[
    f_n = \sum_{l=0}^{M-1} e^{-2\pi i \frac{n}{M}} \hat{f}_l
\]

Then,

\[
    \frac{1}{M} \sum_{n=0}^{M-1} e^{2\pi i \frac{nQ}{M}} f_{(nQ) \mod M} = \hat{f}_{\left[\frac{Q-1}{M}\right]} \mod M
\]

where \( 0 < \left[\frac{Q}{M}\right]^{-1} < M \) is the unique integer such that \( \left[\frac{Q}{M}\right]^{-1} Q \mod M = 1 \mod M \).

Proof. Consider

\[
    \sum_{n=0}^{M-1} e^{2\pi i \frac{nQ}{M}} f_{(nQ) \mod M} = \sum_{l=0}^{M-1} \left( \sum_{n=0}^{M-1} e^{2\pi i \frac{n}{M}} f_{(nQ) \mod M} \right) \hat{f}_l = \sum_{l=0}^{M-1} \left( \sum_{n=0}^{M-1} e^{2\pi i \frac{n}{M}} f_{\left[ \frac{Q}{M} \right]^l \mod M - l} \right) \hat{f}_l
\]
and since \( n \rightarrow nQ \mod M \) is an isomorphism (by lemma 2)

\[
\sum_{n=0}^{M-1} e^{2\pi i \frac{nQ}{M} (m[Q]^{-1}_M \mod M-l)} = M \delta_{m[Q]^{-1}_M \mod M,l},
\]

which implies that

\[
\sum_{n=0}^{M-1} e^{2\pi i \frac{nQ}{M} f(nQ) \mod M} = \hat{f}(m[Q]^{-1}_M) \mod M
\]
as claimed.

**Lemma 4.** Let \( M \in \mathbb{N}/\{0\} \) have the following prime number factorization:

\[
M = \prod_{i=0}^{f} p_i^\mu_i,
\]

where \( \{p_i\}_{i=0}^{f} \) are prime numbers and \( 0 \leq \mu_i \in \mathbb{N} \). Then, every number \( n = \{1, 2, ..., M-1\} \) can be written as

\[
n = m \prod_{i=0}^{f} p_i^{\nu_i},
\]

where \( m \perp M \) and \( m < \prod_{i=0}^{M} p_i^{\nu_i} \) for a unique combination of exponent \( 0 \leq \nu_i \leq \mu_i \forall i \).

**Proof.** This is a simple consequence of the uniqueness of the prime factorization for positive integers.

**Lemma 5.** Let \( M \in \mathbb{N}/\{0\} \) and let \( Q \) be a uniform random variable over \( \mathbb{Q}(M) \). Then,

\[
P(\lvert jQ \mod M \rvert < C) = O\left(\frac{C}{M}\right)
\]

for all \( 0 \leq j < M \) (up to a \( \log(\log(M)) \) factors).

**Proof.** First, from Lemma 4, every integer \( n = \{1, 2, ..., M-1\} \) can be written as

\[
n = m_n \prod_{i=0}^{f} p_i^{\nu_i} = m_n \gamma(\nu),
\]

where \( 0 \leq \nu_i \leq \mu_i \) for all \( i \) and \( m_n \perp M \) (\( M \) has the factorization introduced in Lemma 4), and \( \perp \) indicates coprime. Consider each family of \( \nu \)-tuples independently. Let \( \mathcal{D} \) be the set

\[
\mathcal{D} := \{(\nu_0, \nu_1, ..., \nu_I) : 0 \leq \nu_i \leq \mu_i \forall \ 0 \leq i \leq I\}.
\]

For each element of \( \nu \in \mathcal{D} \), consider the set

\[
s(\nu) := \left\{ 0 < n < M : n \in \mathbb{Z} \text{ and } n = m \prod_{i=0}^{f} p_i^{\nu_i} = m \gamma(\nu) \text{ for some } m \perp M \text{ such that } m < \frac{M}{\gamma(\nu)} \right\}.
\]
Now consider \( n \in s(\nu) \), and

\[
\left( \frac{e^{\zeta} \log(\log(N))}{M} \right) \sum_{q} 1_{qn \mod M = k}(q) \leq \mathbb{P} (nQ \mod M = k) \leq \left( \frac{e^{\zeta} \log(\log(N))}{M} + \frac{3}{\log(\log(N))} \right) \sum_{q} 1_{qn \mod M = k}(q)
\]

following bounds on the Euler totient function [30], where \( \zeta \) is the Euler-Mascheroni constant, and since \( Q \) is uniformly distributed in \( Q \). We now show that the quantity

\[
\sum_{q} 1_{qn \mod M = k}(q)
\]

is bounded by \( \gamma(\nu) = \gcd(n, M) \). To see this, first note that this quantity corresponds to the number of integers \( q \) which hash to the integer \( k \) through the map \( q \to (nq) \mod M \). Now, assume there exists some \( q \) such that

\[
nq \mod M = k,
\]

which implies that

\[
nq + iM = k
\]

for some integer \( i \in \mathbb{Z} \). This is a Diophantine equation which has infinitely many solutions if and only if \( \gcd(n, M) = \gamma(\nu) \) divides \( k \). Otherwise, it has no solution. Assuming it does and \( q_0 \) is a particular solution, all solutions take the form

\[
q = q_0 + u \frac{M}{\gamma(\nu)},
\]

where \( u \in \mathbb{Z} \). However, since \( 0 \leq q < M \) the number of possible solutions must be such that,

\[
\gamma(\nu) - 1 \leq \# \left\{ q \in [0, M) : q = q_0 + u \frac{M}{\gamma(\nu)} \right\} \leq \gamma(\nu).
\]

Thus,

\[
e^{\zeta} \log(\log(N)) \frac{\gamma(\nu) - 1}{M} \leq \mathbb{P} (nQ \mod M = k) \leq \left( e^{\zeta} \log(\log(N)) + \frac{3}{\log(\log(N))} \right) \frac{\gamma(\nu)}{M}
\]

for all \( n \in s(\nu) \).

We now treat: \( \mathbb{P} (|nQ \mod M| \leq C) \). Before we proceed however, recall that Eq. (15) has a solution (for \( n \in s(\nu) \)) if and only if \( \gamma(\nu)|k \). We then write,

\[
\mathbb{P} (|nQ \mod M| \leq C) = \sum_{|k| \leq C} 1_{\gamma(\nu)|k}(k) \mathbb{P} (nQ \mod M = k)
\]

from which it follows that,

\[
\mathbb{P} (|nQ \mod M| \leq C) \leq \left( e^{\zeta} \log(\log(M)) + \frac{3}{\log(\log(M))} \right) \frac{\gamma(\nu)}{M} \sum_{|k| \leq C} 1_{\gamma(\nu)|k}(k)
\]

\[
\leq \left( e^{\zeta} \log(\log(M)) + \frac{3}{\log(\log(M))} \right) \frac{\gamma(\nu)}{M} \left\lfloor \frac{C}{\gamma(\nu)} \right\rfloor
\]

\[
\leq \left( e^{\zeta} \log(\log(M)) + \frac{3}{\log(\log(M))} \right) \frac{C}{\gamma(\nu)}
\]

26
since there can be at most \( \left\lceil \frac{C}{\gamma(\nu)} \right\rceil \) integers in \( |k| \leq C \) that are divisible by \( \gamma(\nu) \). This is the desired result. □

**Lemma 6.** Consider a function \( f(x) \) of the form of Eq. (1) and satisfying the constraint Eq. (2). Let \( 0 < \sigma, 0 < \delta < 1 \) and

\[
\Delta = \frac{\|\hat{f}\|_1}{\mu},
\quad
\mu = \min_{j \in S} \hat{f}_j,
\quad
K = O \left( \sigma \sqrt{\log \left( \frac{2 \Delta}{\delta} \right)} \right).
\]

Finally, let \( F_K(\cdot) \) and \( \Psi_\sigma(\cdot) \) be the operators defined in Eq. (3) and Eq. (6) respectively. Then,

\[
\left[ F_K \left( \Psi_\sigma \left( \{f_k; M\}_{k=0}^{M-1} \right) \right) \right]_n \geq \delta \mu
\]

implies that

\[
\inf_{j \in S} \left| n - \frac{j}{K} \right| \geq \sigma \sqrt{\log \left( \frac{2 \Delta}{\delta} \right)}
\]

and,

\[
\inf_{j \in S} \left| n - \frac{j}{K} \right| \leq \sigma \sqrt{\log \left( \frac{\Delta}{\delta} \right)}.
\]

implies that

\[
\left[ F_K \left( \Psi_\sigma \left( \{f_k; M\}_{k=0}^{M-1} \right) \right) \right]_n \geq \delta \mu. \tag{16}
\]

**Proof.** Consider the quantity

\[
\phi_n = \left[ F_K \left( \Psi_\sigma \left( \{f_k; M\}_{k=0}^{M-1} \right) \right) \right]_n = \frac{1}{K} \sum_{k=0}^{K-1} e^{2\pi i \frac{n k}{K}} \hat{g}_\sigma \left( \frac{k}{K} \right) f_k; M
\]

\[
= \sum_{j \in S} \left( \frac{1}{K} \sum_{k=0}^{K-1} e^{2\pi i \frac{n - j}{K} \frac{k}{K}} \hat{g}_\sigma \left( \frac{k}{K} \right) \right) \hat{f}_j \tag{17}
\]

We recognize the factor in parentheses in Eq. (17) as the inverse DFT of \( e^{-2\pi i \frac{n k}{K}} \hat{g}_\sigma (\cdot) \) evaluated at \( n \). Following the decay properties of the Gaussian filter in phase-space, it follows that if,

\[
K = O \left( \sigma \sqrt{\log \left( \frac{2 \Delta}{\delta} \right)} \right)
\]

then

\[
\frac{1}{K} \sum_{k=0}^{K-1} e^{2\pi i \frac{n - j}{K} \frac{k}{K}} \hat{g}_\sigma \left( \frac{k}{K} \right) = e^{-\frac{|n-j|^2}{2 \sigma^2}} + \epsilon_j
\]

27
where: \( \sqrt{\sum_j |\epsilon_j|^2} \leq \frac{\delta}{2\Delta} \) for all \( j \in S \) so that

\[
\phi_n = \sum_{j \in S} e^{-\frac{|n-jK|^2}{\sigma^2}} \hat{f}_j + \sum_{j \in S} \epsilon_j \hat{f}_j
\]

and

\[
\left| \sum_{j \in S} \epsilon_j \hat{f}_j \right| \leq \|\epsilon\|_\infty \|\hat{f}\|_1 \leq \frac{\delta}{2\Delta} \|\hat{f}\|_1
\]

by Hölder inequality.

Now assume: \( |\phi_n| \geq \delta \mu \). This implies that

\[
\sum_{j \in S} e^{-\frac{|n-jK|^2}{\sigma^2}} \hat{f}_j \geq \frac{\delta \Delta}{|\hat{f}|_1} - \frac{\delta}{2\Delta} \|\hat{f}\|_1 = \frac{\delta}{2\Delta} \|\hat{f}\|_1 = \frac{\delta \mu}{2}.
\]

We claim that this cannot occur unless,

\[
\inf_{j \in S} |n - j\frac{K}{M}| \leq \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right)}.
\]

(19)

We proceed by contradiction. Assume the opposite holds. Then,

\[
\sum_{j \in S} e^{-\frac{|n-jK|^2}{\sigma^2}} \hat{f}_j \leq \|\hat{f}\|_1 e^{-\frac{\inf_{j \in S} |n-jK|^2}{\sigma^2}} < \frac{\delta}{2\Delta} \|\hat{f}\|_1 = \frac{\delta \mu}{2}
\]

thanks to the positivity of the elements of \( \hat{f} \). This is a contradiction. Thus, Eq. (19) must indeed hold. This proves the first part of the proposition.

For the second part, assume

\[
\inf_{j \in S} |n - j\frac{K}{M}| \leq \sigma \sqrt{\log \left( \frac{\Delta}{\delta} \right)}
\]

(20)

holds. Letting \( j^* \) be such that \( |n - j^*\frac{K}{M}| = \inf_{j \in S} |n - j\frac{K}{M}| \), we note that,

\[
\sum_{j \in S} e^{-\frac{|n-jK|^2}{\sigma^2}} \hat{f}_j \geq e^{-\frac{|n-j^*K|^2}{\sigma^2}} \hat{f}_{j^*} \geq \frac{\delta}{\Delta} \|\hat{f}\|_1 = \delta \mu
\]

since \( \hat{f} \) and the Gaussian are all positive. This shows the second part.

We are now ready to prove the validity of the Algorithm.

**Proposition 4.** Consider a function \( f(x) \) of the form of Eq. (1) and satisfying the constraint Eq. (2), and let \( \Pi_Q(\cdot) \), \( \Psi_{\sigma}(\cdot) \) and \( F_K(\cdot) \) be the operators defined in Eq. (4), Eq. (6) and Eq. (3) respectively where \( \delta, K, \Delta \) and \( \mu \) are as in Lemma 4 and

\[
\sigma = \frac{\alpha}{\sqrt{\log \left( \frac{2\Delta}{\delta} \right)}}
\]

28
for some \(0 < \alpha < 1\). Assume further that the integers \(\{Q^{(l)}\}_{l=1}^{L}\) are chosen independently and uniformly at random within \(Q(M)\), and that \(M\) is such that \(\frac{K}{2M} \leq \sigma \sqrt{\log \left(\frac{2\Delta}{\delta}\right)}\). Consider

\[
\phi_{n}(l) := \left[ F_{K} \left( \Psi_{\sigma} \left( \Pi_{Q^{(l)}} \left\{ f \left( \frac{k}{M} \right) \right\}_{k=0}^{M-1} \right) \right) \right]_{n}
\]

(21)

Then,

\[
P\left( \bigcap_{l=1}^{L} \left\{ |\phi_{\chi_{\mathcal{K}}(j)}(l)| \geq \delta \mu \right\} \right) \leq \alpha^{L}
\]

for all \(j \notin \mathcal{S}\), where \(\chi_{\mathcal{K}}(\cdot)\) is defined in Definition 4 and

\[
|\phi_{\chi_{\mathcal{K}}(j)}(l)| \geq \delta \mu
\]

almost surely for all \(j \in \mathcal{S}\).

Proof. First, consider \(i \notin \mathcal{S}\) and fix \(l\). From independence, the probability in Eq. (22) is equal to,

\[
\prod_{l=1}^{L} P\left( |\phi_{\chi_{\mathcal{K}}(j)}(l)| \geq \delta \mu \right).
\]

So it is sufficient to consider a fixed \(l\). As a consequence of Lemma 6 and Lemma 3 we have the inclusion,

\[
\left\{ |\phi_{\chi_{\mathcal{K}}(j)}(l)| \geq \delta \mu \right\} \subset \left\{ \inf_{j \in \mathcal{S}} \left| \left( i \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod \frac{M}{\mathcal{K}} - \left( j \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod \frac{M}{\mathcal{K}} \right| \leq \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right)} \right\}
\]

\[
\subset \left\{ \inf_{j \in \mathcal{S}} \left| \left( i \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod \frac{M}{\mathcal{K}} - \left( j \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod \frac{M}{\mathcal{K}} \right| \leq \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right) + K} \right\}
\]

\[
\subset \bigcup_{j \in \mathcal{S}} \left\{ \left| \left( (i-j) \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod \frac{M}{\mathcal{K}} \right| \leq \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right) + K} \right\},
\]

which implies that the probability of the set \(\left\{ |\phi_{\chi_{\mathcal{K}}(j)}(l)| \geq \delta \mu \right\}\) for each fixed \(l\) is bounded by

\[
\sum_{j \in \mathcal{S}} P\left( \left| \left( (i-j) \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod M \right| \leq \frac{M}{K} \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right) + \frac{1}{2}} \right) \leq \mathcal{O} \left( R \left( \frac{M}{K} \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} + \frac{1}{2} \right) \left( \frac{2\Delta}{\delta} \right)} \right) \right) = \mathcal{O}(\alpha),
\]

by the union bound. However, since \(i \notin \mathcal{S}\) and \(j \in \mathcal{S}\) \((i-j) \neq 0\) and it follows by Lemma 5 that

\[
\sum_{j \in \mathcal{S}} P\left( \left| \left( (i-j) \left[ Q^{(l)} \right]_{M}^{-1} \right) \bmod M \right| \leq \frac{M}{K} \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} \right) + \frac{1}{2}} \right) \leq \mathcal{O} \left( R \left( \frac{M}{K} \sigma \sqrt{\log \left( \frac{2\Delta}{\delta} + \frac{1}{2} \right) \left( \frac{2\Delta}{\delta} \right)} \right) \right) = \mathcal{O}(\alpha),
\]

since \(K \geq R\) and by definition of \(\sigma\). Therefore,

\[
P\left( \bigcap_{l=1}^{L} \left\{ |\phi_{\chi_{\mathcal{K}}(j)}| \geq \delta \mu \right\} \right) \leq \mathcal{O}(\alpha^{L})
\]

29
as claimed.

As for the second part of the proposition, note that if \( i \in \mathcal{S} \) then

\[
\inf_{j \in \mathcal{S}} \left| \left( \frac{(i [Q^{(l)}]_{\mathcal{M}}^{-1})}{M} \right) \mod \frac{M}{K} - \left( \frac{(j [Q^{(l)}]_{\mathcal{M}}^{-1})}{M} \right) \mod \frac{M}{K} \right| \leq \inf_{j \in \mathcal{S}} \left( \frac{(i [Q^{(l)}]_{\mathcal{M}}^{-1})}{M} \mod \frac{M}{K} - \left( \frac{(j [Q^{(l)}]_{\mathcal{M}}^{-1})}{M} \mod \frac{M}{K} \right) \right) + \frac{K}{2M}
\]

\[
= \frac{K}{2M}
\]

\[
\leq \sigma \sqrt{\log \left( \frac{\Delta}{\delta} \right)}
\]

by assumption. By Lemma 6, this implies that

\[
|\phi_{X_{\mathcal{M}}}^{(j)}(l)| \geq \delta \mu
\]

and since this is true regardless of the value of the random variable \( Q^{(l)} \), we conclude that it holds almost surely.

\[\Box\]

### A.2 Proofs of Section 3.3

**Lemma 7.** Let \( \{P^{(t)}\}_{t=1}^{T} \) be prime numbers greater than or equal to \( 0 < R \in \mathbb{N} \), and let \( Q^{(t)} \) belong to \( Q(P^{(t)}) \). Further let \( x_i, x_j \in \{0, 1, \ldots, N - 1\} \) such that,

\[x_i Q^{(t)} \mod P^{(t)} = x_j Q^{(t)} \mod P^{(t)}, \ t = 1, 2, \ldots, T.\]

If \( T > \log_R(N) \), then \( x_i = x_j \).

**Proof.** Consider \( \{P^{(t)}\} \) as described and \( T > \log_R(N) \), and assume that

\[x_j Q^{(t)} \mod P^{(t)} = x_i Q^{(t)} \mod P^{(t)}\]

for \( t = 0, 1, \ldots, T \). Since \( Q^{(t)} \perp P^{(t)} \) this is an isomorphism (Lemma 2) and therefore the above statement is equivalent to

\[x_j \mod P^{(t)} = x_i \mod P^{(t)}\]

for \( t = 0, 1, \ldots, T \). This implies in particular that

\[P^{(t)} \mid (x_j - x_i)\]

for \( t = 0, 1, \ldots, T \), and that

\[\text{lcm}\{\{P^{(t)}\}_{t=1}^{T}\} \mid (x_j - x_i).\]

However, since the integers \( \{P^{(t)}\} \) are prime (and therefore coprime),

\[\text{lcm}(P^{(t)}) = \prod_{t} P^{(t)} \geq \left( \min_{t} P^{(t)} \right)^T \geq R^{\log_R(N)} = N.\]

Therefore, since \( x_j \neq x_i \),

\[|x_j - x_i| \geq N,\]

which is a contradiction since both belong to \( \{0, 1, \ldots, N - 1\} \).

\[\Box\]
Therefore, \( P_{0} \) random variables uniformly distributed in \( F \), since Proposition 5.

Let \( \tau(t) \) be defined as in Eq. (23) and independence, the expectation can be written as, \( (FB)^{\ast}(FB) = B^{\ast}F^{\ast}FB = B^{\ast}B \),

\[
[B^{(t)}B^{(t)}]_{ij} = \sum_{s} B_{si}^{(t)} B_{sj}^{(t)} = \begin{cases} 
1 & \text{if } x_{i}[Q^{(t)}]_{P(t)}^{-1} \mod P(t) = x_{j}[Q^{(t)}]_{P(t)}^{-1} \mod P(t) \text{ if } s \neq t \\
0 & \text{o.w. if } s = t
\end{cases}
\]

Therefore, \( \mathbb{P} \left( \| (I - (FB)^{\ast}(FB))x \|_{2} > \frac{1}{2} \right) \leq 4 \mathbb{E} \left[ \sum_{i \neq j} \frac{1}{T^{2}} \sum_{t} \bar{x}_{i} [B^{(t)}^{\ast}B^{(t)}]_{ij} x_{j} \right]^{2} \)

by Chebyshev inequality. Furthermore, thanks to Eq. (23) and independence, the expectation can be written as,

\[
\mathbb{E} \left[ [B^{(t)}^{\ast}B^{(t)}]_{ij} [B^{(s)}^{\ast}B^{(s)}]_{kl} \right] = \begin{cases} 
\mathbb{P} \left( \{ (i - j)[Q^{(t)}]_{P(t)}^{-1} \mod P(t) = 0 \} \right) \mathbb{P} \left( \{ (k - l)[Q^{(s)}]_{P(s)}^{-1} \mod P(t) = 0 \} \right) & \text{if } s \neq t \\
\mathbb{P} \left( \{ (i - j)[Q^{(t)}]_{P(t)}^{-1} \mod P(t) = 0 \} \cap \{ (k - l)[Q^{(s)}]_{P(s)}^{-1} \mod P(t) = 0 \} \right) & \text{if } s = t
\end{cases}
\]

Now, let \( \tau(i, j) \) be defined as \( \tau(i, j) := \{ P(t) : i[Q^{(t)}]_{P(t)}^{-1} \mod P(t) = j[Q^{(t)}]_{P(t)}^{-1} \mod P(t) \} \).

The case \( s \neq t \) is treated as follows,

\[
\mathbb{P} \left( \{ (i - j)[Q^{(t)}]_{P(t)}^{-1} \mod P(t) = 0 \} \mid P(t) \in \tau(i, j) \right) \mathbb{P} \left( P(s) \in \tau(k, l) \right) \leq O \left( \frac{\log^2(N) \log(\log(R))}{2R} \right)
\]

where \( \tau(k, l) = \#\tau(k, l) \) and \( \#\tau(i, j) \).
because $P^{(t)} > R$ and is uniformly distributed within a set of cardinality $2R$, where the bound on 
$\#\tau(k, l)$ follows from Lemma 7 and because the probability

$$P \left( \left\{ (k - l)[Q^{(t)}]^{-1}_{P^{(t)}} \mod P^{(s)} = 0 \right\} \mid P^{(s)} = P_2 \right) \leq O \left( \frac{\log(\log(R))}{R} \right)$$

by Lemma 8 following the fact that $Q^{(t)}$ are i.i.d and uniformly distributed in $Q(P^{(t)})$ and co-prime to $P^{(t)}$.

This leaves us the case $s = t$. To this purpose, we further split this case into two subcases: that when $i - j = k - l$ and that when $i - j \neq k - l$. When

$$i - j = k - l$$

we obtain,

$$\sum_{s, t = 1}^{T} P \left( \{ s = t \} \cap \{ i - j = k - l \} \cap \left\{ (i - j)[Q^{(t)}]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right\} \cap \left\{ (k - l)[Q^{(s)}]^{-1}_{P^{(s)}} \mod P^{(s)} = 0 \right\} \right)$$

$$= P \left( \left\{ (k - l)[Q^{(t)}]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right\} \mid P^{(t)} \in \tau(k, l) \right) P^{(t)} \in \tau(k, l)$$

$$\leq O \left( \frac{\log(\log(R))}{R} \right) \frac{\#\tau(k, l)}{2R}$$

$$\leq O \left( \frac{\log_R(N) \log(\log(R))}{2R^2} \right)$$

since $k \neq l$, following an argument similar to the previous one.

This leaves the case $s = t, i - j \neq k - l$. To this purpose, we note that,

$$P \left( \{ s = t \} \cap \left\{ (i - j)[Q]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right\} \cap \left\{ (k - l)[Q]^{-1}_{P^{(s)}} \mod P^{(s)} = 0 \right\} \right) \leq P \left( (i - j - k + l)[Q]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right)$$

which implies that,

$$P \left( \{ s = t \} \cap \{ i - j \neq k - l \} \cap \left\{ (i - j)[Q^{(t)}]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right\} \cap \left\{ (k - l)[Q^{(s)}]^{-1}_{P^{(s)}} \mod P^{(s)} = 0 \right\} \right)$$

$$\leq P \left( \{ i - j \neq k - l \} \cap \{ i - j - k + l \}[Q^{(t)}]^{-1}_{P^{(t)}} \mod P^{(t)} = 0 \right) \mid P^{(t)} \in \tau(i - j, k - l) \right) P^{(t)} \in \tau(i - j, k - l)$$

$$\leq O \left( \frac{\log(\log(R))}{R} \right) \frac{\#\tau(i - j, k - l)}{2R}$$

$$\leq O \left( \frac{\log_R(N) \log(\log(R))}{2R^2} \right)$$

since $i - j - k + l \neq 0$, by Lemma 7 and the properties of the distribution of $P^{(t)}$. 32
Putting everything together we find that there exists some positive constant $C$ such that

$$
P \left( \| (I - (FB)^*(FB))x \|_2 > \frac{1}{2} \right) \leq 4C \sum_{i \neq j} \sum_{k \neq l} \bar{x}_i \bar{x}_j \bar{x}_k \frac{1}{T^2} \left[ \sum_{s,t=1}^T \mathbb{E} \left[ \mathbb{I}_{s=t} [B(t)^* B(t)]_{ij} [B(s)^* B(s)]_{kl} \right] + \mathbb{E} \left[ \mathbb{I}_{s=t} \mathbb{I}_{i-j=k-l} [B(t)^* B(t)]_{ij} [B(s)^* B(s)]_{kl} \right] \right. + \left. \mathbb{E} \left[ \mathbb{I}_{s=t} \mathbb{I}_{i-j \neq k-l} [B(t)^* B(t)]_{ij} [B(s)^* B(s)]_{kl} \right] \right] \leq 4C \left( \frac{\log^2(\log(R)) \log(\log(R))^2}{4R^4} \right) \left( \sum_{k \neq l} \bar{x}_l \bar{x}_k \right)^2 + \frac{4C}{T} \left( \frac{\log(R) \log(\log(R))}{2R^2} \right) \left( \sum_{k \neq l} \bar{x}_l \bar{x}_k \right) \left( \sum_j \bar{x}_{j+k-l} x_j \right) + \frac{4C}{T} \left( \frac{\log(R) \log(\log(R))}{2R^2} \right) \left( \sum_{i \neq j} \bar{x}_i \bar{x}_j \right) \left( \sum_{k \neq l} \bar{x}_l x_k \right).
$$

We further note that $\sum_{i \neq j} \bar{x}_i x_k$ is a bilinear form bounded by the norm of an $R \times R$ matrix with all entries equal to 1 except the diagonal which is all zeros. It is easy to work out this norm which is equal to $R - 1$ so that,

$$\frac{1}{R} \sum_{k \neq l} \bar{x}_l x_k < 1$$

Finally, by Cauchy-Schwartz inequality,

$$\left| \sum_j \bar{x}_{j+k-l} x_j \right| \leq \sqrt{\sum_j |x_{j+k-l}|^2} \sqrt{\sum_j |x_j|^2} = \|x\|_2^2 = 1.$$ 

Thus,

$$P \left( \| (I - (FB)^*(FB))x \|_2 > \frac{1}{2} \right) \leq \frac{C \log^2(\log(R)) \log(\log(R))^2}{R^2} + 2C \frac{\log(R) \log(\log(R))}{T} \left( \frac{1}{R} + 1 \right) \leq \frac{1}{2}$$

and thus if

$$R \geq \sqrt{4C \log(R) \log(\log(R))},$$

$$T \geq 4C \log(R) \log(\log(R)),$$

one indeed obtains

$$P \left( \| (I - (FB)^*(FB))x \|_2 > \frac{1}{2} \right) \leq \frac{1}{2}$$

as claimed. \qed
Corollary 2. Under the hypotheses of Proposition 5, the solution to the linear system
\[ FBx = b \]
takes the form,
\[ x = \sum_{n=0}^{\infty} (I - (FB)^* (FB))^n b \]
with probability at least \((1 - p)\).

Proof. By Proposition 5, \( ||I - (FB)^* (FB)||_2 < \frac{1}{2} \) with probability at least \((1 - p)\). When this is the case we write,
\[ FBx = b \Leftrightarrow B^* Bx = B^* F^* b \Leftrightarrow [I - (I - B^* B)] x = B^* F^* b \]
In this case, it is easy to verify that the Neumann series,
\[ x = \sum_{n=0}^{\infty} (I - (FB)^* (FB))^n b, \]
satisfies this last equation, and that the sum converges exponentially fast. \(\Box\)

A.3 Proofs of Section 3.4

Lemma 1. In the noisy case and under the assumptions of Section 3.4, the error term of the computed value \( \phi_n^{(k)}(Q) \) takes the form,
\[ \eta [F_K^* (\Psi_\sigma (\Pi_Q (\nu_{n;M_k})))]_n \]
and is uniformly bounded by an expression on the order of,
\[ ||[F_K^* (\Psi_\sigma (\Pi_Q (\nu_{n;M_k})))]_n||_\infty = O(\eta) \]

Proof. First, not that since \( \Pi_Q(\cdot) \) is an isomorphic permutation operator (for all \( Q \in Q(M_k) \)) one has
\[ ||\Pi_Q||_\infty = 1. \]
Similarly, since the filtering operator \( \Psi_\sigma(\cdot) \) is diagonal with nonzero entries \( \hat{g}_\sigma(n) \), then
\[ ||\Psi_\sigma||_\infty = \sup_{n \in \mathbb{Z}} |\hat{g}_\sigma(n)| \leq \sqrt{\pi} \sigma < \infty. \]
Finally, since \( F_K^* \) is a size-\( K \) DFT, we get from the Hausdorff-Young inequality [4] that
\[ ||F_K^* g||_\infty \leq ||g||_1 \]
for all \( g \in \ell_1 \). Putting everything together results in
\[ ||[F_K^* (\Psi_\sigma (\Pi_Q (\nu_{n;M_k})))]_n||_\infty \leq \eta \sqrt{\pi} \sigma ||\nu||_1 = O(\eta) \]
since \( ||\nu||_1 = O(1) \) by assumption. \(\Box\)
A.4 Proof of Section 4

Proposition 3. Assume the function \( f : [0,1)^d \to \mathbb{C} \) is bandlimited and periodic and has form (12). Then,

\[
\int_{[0,1)^d} e^{-2\pi i j \cdot x} f(x) \, dx = \frac{1}{N} \sum_{n=0}^{N-1} e^{-2\pi i j \cdot x_n} f(x_n)
\]

for all \( j \in \{0,1,\ldots,M-1\} \), where,

\[
x_n = \frac{ng \mod N}{N}
\]

and \( g = (1,M,M^2,\ldots,M^{d-1}) \) and \( N = M^d \).

Proof. First, note that

\[
\int_{[0,1]^d} f(x) \, dx = \hat{f}_0.
\]

Then, substitute the samples in the quadrature to obtain

\[
\frac{1}{N} \sum_{n=0}^{N-1} e^{-2\pi i k \cdot x_n} f(x_n) = \frac{1}{N} \sum_{n=0}^{N-1} e^{-2\pi i k \cdot \frac{ng \mod N}{N}} \left( \sum_{k \in [0,M)^d} \hat{f}_k e^{2\pi i k \cdot \frac{ng \mod N}{N}} \right)
\]

since \( e^{2\pi i k \cdot \frac{ng \mod N}{N}} = e^{2\pi i k \cdot \frac{ng}{N}} \). Note however that

\[
\frac{1}{N} \sum_{n=0}^{N-1} e^{-2\pi i \frac{n(k \cdot g)}{N}} = D_N(k \cdot g),
\]

which is the Dirichlet kernel and is equal to 0 unless \( k \cdot g = 0 \mod N \), in which case it is equal to 1. Thus,

\[
\frac{1}{N} \sum_{n=0}^{N-1} f(x_n) = \hat{f}_0 + \sum_{k \in [0,M)^d \setminus 2^d} \hat{f}_k.
\]

In order to show that the quadrature is exact, it therefore suffices to show that the remaining sum on the right-hand side of the previous equation is trivial. To see this, consider

\[
|k \cdot g| = |k_1 + k_2 M + \ldots + k_{d-1} M^{d-1}| \leq M \sum_{i=0}^{d-1} M^i = M \frac{1 - M^d}{1 - M} < M^d = N,
\]

where the inequality is strict for any finite \( 1 < M \in \mathbb{N} \). This implies that there cannot be any \( k \) other than 0 in the domain of interest such that \( k \cdot g \mod N \equiv 0 \). This sum is therefore empty and the result follows.
References

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

[2] Adi Akavia. Deterministic sparse Fourier approximation via approximating arithmetic progressions. *Information Theory, IEEE Transactions on*, 60(3):1733–1741, 2014.

[3] Adi Akavia, Shafi Goldwasser, and Shmuel Safra. Proving hard-core predicates using list decoding. In *FOCS*, volume 44, pages 146–159, 2003.

[4] William Beckner. Inequalities in Fourier analysis. *Annals of Mathematics*, pages 159–182, 1975.

[5] Petros Boufounos, Volkan Cevher, Anna C Gilbert, Yi Li, and Martin J Strauss. What’s the frequency, Kenneth?: Sublinear Fourier sampling off the grid. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 61–72. Springer, 2012.

[6] Eleanor Chu and Alan George. *Inside the FFT black box: serial and parallel Fast Fourier Transform algorithms*. CRC Press, 1999.

[7] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[8] Tom Darden, Darrin York, and Lee Pedersen. Particle mesh ewald: An $n \log(n)$ method for ewald sums in large systems. *The Journal of chemical physics*, 98(12):10089–10092, 1993.

[9] Matteo Frigo and Steven G Johnson. FTFW: An adaptive software architecture for the FFT. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 3, pages 1381–1384. IEEE, 1998.

[10] Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Erik Price, and Lixin Shi. Sample-optimal average-case sparse Fourier Transform in two dimensions. In *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, pages 1258–1265. IEEE, 2013.

[11] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 152–161. ACM, 2002.

[12] Anna C Gilbert, S Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Optics & Photonics 2005*, pages 59141A–59141A. International Society for Optics and Photonics, 2005.

[13] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[14] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse Fourier Transform. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, pages 563–578. ACM, 2012.
[15] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier Transform. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1183–1194. SIAM, 2012.

[16] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on, pages 514–523. IEEE, 2014.

[17] Piotr Indyk and Michael Kapralov. Sparse Fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. 2014.

[18] Piotr Indyk, Michael Kapralov, and Eric Price. (nearly) sample-optimal sparse Fourier Transform. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 480–499. SIAM, 2014.

[19] Mark A Iwen. Combinatorial sublinear-time Fourier algorithms. Foundations of Computational Mathematics, 10(3):303–338, 2010.

[20] Mark A Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. Applied And Computational Harmonic Analysis, 34(1):57–82, 2013.

[21] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. SIAM Journal on Computing, 22(6):1331–1348, 1993.

[22] David Lawlor, Yang Wang, and Andrew Christlieb. Adaptive sub-linear time Fourier algorithms. Advances in Adaptive Data Analysis, 5(01):1350003, 2013.

[23] Juan M Lopez-Sanchez and Joaquim Fortuny-Guasch. 3-D radar imaging using range migration techniques. Antennas and Propagation, IEEE Transactions on, 48(5):728–737, 2000.

[24] Stéphane Mallat. A wavelet tour of signal processing. Academic press, 1999.

[25] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. SIAM Journal on Computing, 24(2):357–368, 1995.

[26] Jacques Morgenstern. Note on a lower bound on the linear complexity of the Fast Fourier Transform. Journal of the ACM (JACM), 20(2):305–306, 1973.

[27] Sanjay Pawar and Kannan Ramchandran. Computing a k-sparse n-length discrete Fourier Transform using at most 4k samples and o (k log k) complexity. In Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on, pages 464–468. IEEE, 2013.

[28] Gerlind Plonka and Katrin Wannenwetsch. Deterministic sparse FFT algorithms. PAMM, 15(1):667–668, 2015.

[29] Gerlind Plonka and Katrin Wannenwetsch. A sparse fast Fourier algorithm for real nonnegative vectors. arXiv preprint arXiv:1602.05444, 2016.

[30] Paulo Ribenboim. The book of prime number records. Springer Science & Business Media, 2012.

[31] Charles Van Loan. Computational frameworks for the Fast Fourier Transform, volume 10. Siam, 1992.