SUB-LINEAR SPARSE FOURIER TRANSFORM ALGORITHM

By

Ruochuan Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Applied Mathematics — Doctor of Philosophy

2017

ABSTRACT

SUB-LINEAR SPARSE FOURIER TRANSFORM ALGORITHM

By

Ruochuan Zhang

The Discrete Fourier Transform (DFT) plays a crucial role in signal processing and scientific computing. The most famous algorithm of computing the DFT is the Fast Fourier Transform (FFT), which has runtime $\mathcal{O}(N \log N)$ for an input vector with length N. However, with the increasing size of data set, the FFT is no longer fast enough and often becomes the major computational bottleneck in many applications. The Sparse Fourier Transform (SFT) tries to solve this problem by finding the best $s-$term Fourier representation using only a subset of the input data, in time sub-linear in the data set size $\mathcal{O}(poly(s, \log N))$. Some of the existing SFT algorithms are capable of working with equally spaced samples[23], while others just assume that the algorithms can sample anywhere they want[4, 6], which is an unrealistic assumption in many real-world applications. In this thesis, we propose a generic method of transforming any noise robust SFT algorithm into a sublinear-time sparse DFT algorithm which rapidly approximates $F\mathbf{f}$ from a given input vector $\mathbf{f} \in \mathbb{C}^N$, where F is the DFT matrix. Our approach is based on filter function and fast discrete convolution. We prove that with an appropriate filter function g (periodic Gaussian function in this thesis), one can always approximate the value of the convolution function $g * f$ at the desired point rapidly and accurately even when f is a high oscillating function. We then construct several new sublinear-time sparse DFT algorithms from existing sparse Fourier algorithms which utilize unequally spaced function samples [6, 4, 8]. Besides giving the theoretical runtime and error guarantee, we also show empirically that the best of these new discrete SFT algorithms outperforms both FFTW[24] and sFFT2.0[23] in the sense of runtime and robustness when the vector length N is large. At the end of the thesis, we present a deterministic sparse Fourier transform algorithm which breaks the quadratic-in-sparsity runtime bottleneck for a large class of periodic functions exhibiting structured frequency support. We show empirically that this structured SFT algorithm outperforms standard sparse Fourier transforms in the rapid recovery of block frequency sparse functions.

To my family

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Mark Iwen and my mentor Prof. Andrew Christlieb, for their continuous support of my Ph.D. study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Zhengfang Zhou and Prof. Matthew Hirn, for their insightful comments and encouragement. I would like to thank Prof. Zhengfang Zhou in particularly, for giving me the opportunity to study at MSU as a graduate student. Without his precious support, it would not be possible to conduct my study. My sincere thanks also go to Prof. Peiru Wu, who open the industrial gate for me and provided me an opportunity to work in close collaboration with people from industry. Beyond her scientific guidance, her advice on navigating an industry career has been invaluable.

I would like to thank the support stuff of Department of Mathematics for all their help with administrative matters. I would like to thank my fellow math students for their friendship over the past five years, for the sleepless nights we were working together before deadlines, and for all the fun we have had. They made the experience of graduate school more rewarding.

Many thanks go to my parents for supporting me spiritually throughout writing this thesis and my life in general. Last but not the least, I would like to express my profound gratitude to my partner Xiaojuan and my son Seth, who have shown me the deepest love, kindness, and patience throughout our time together. This dissertation stands as a testament to your unconditional love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Sparse Fourier Transform

The Discrete Fourier Transform (DFT) is one of the most important components of many computational techniques. It has been widely used in signal processing and scientific computing. The discrete Fourier transform converts a finite sequence of equally-spaced samples of a function in time space into an equivalent-length sequence of equally-spaced samples of the function in Fourier space. The most popular approach for computing the DFT is the Fast Fourier Transform (FFT). It was invented by J. W. Cooley and J. W. Tukey in 1965 [17]. Without directly applying the definition of DFT, the FFT rapidly computes such transformations by factoring the DFT matrix into a product of sparse (mostly zero) factors [18], where the DFT matrix $F \in \mathbb{C}^{N\times N}$ is defined as

$$F_{\omega,j} := \frac{e^{-2\pi \mathbb{i}\omega j/N}}{N} \tag{1.1}$$

for $0 \le \omega, j < N$. As a result, it reduces the runtime complexity of computing a DFT of a length N sequence from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, which represented a major leap forward in the size of problems that could be solved on available hardware. The FFT was described as "the most important numerical algorithm of our lifetime" by Gilbert Strang [20], and it has been named one of the "Top Ten Algorithms" of the past century by the IEEE journal

Computing in Science & Engineering [19]. However, in recent years with the concept of Big Data becoming more and more popular, the size of the data set can easily exceed terabytes. Even with advanced hardware and optimized implementations (e.g. parallel computing and GPU), for large N the FFT still represents the major computational bottleneck in many applications. Furthermore, by the Nyquist−Shannon sampling theorem [21], one needs at least N equally-spaced samples to recover a signal with bandwidth N, which may cause some potential sampling problems. Since when N is very large, it can be challenging to acquire enough data to run the FFT algorithm. For example, patients need to spend a long time in an MRI machine because it needs a lot of samples to retrieve the medical image. Because of above reasons, people try to develop algorithms that can compute the Fourier Transform in sub-linear time (the runtime is considerably smaller than the size of the data size N) and use a minuscule fraction of the input data. The Sparse Fourier Transform (SFT) provides exactly these desired features.

One of the reasons the FFT cannot satisfy the sub-linear runtime requirement is because it computes all N Fourier coefficients. However, it is well known that in many applications (e.g. video, audio, medical images, spectroscopic measurements, GPS signals, seismic data, etc.) the DFT of the signals are compressible or are sparse, i.e. only s frequencies are non-zeros or are significantly large. In this case, when $s \ll N$, one can retrieve the information with high accuracy using only the coefficients of the s most significant frequencies. The SFT uses the same strategy by asking the SFT algorithm to only report the largest s terms in the signal's DFT. Because of that, it is possible for the SFT to significantly outperform even highly optimized FFT implementations [24] when the signal is sparse in Fourier space [23, 22, 2, 4, 6].

To achieve sub-linear runtime, besides only reporting significant frequencies, the SFT cannot even afford to read all the inputs (which takes $\mathcal{O}(N)$ runtime complexity for a signal with bandwidth N). So all the SFT algorithms must either be approximate or succeed with high probability. The discrete uncertainty principle proved by Dohono and Stark [16] in 1989 provides the theoretical guarantee for sub-sampling in the time domain if the signal is sparse in Fourier space (actually, Fourier space is one possible choice, however, in this paper, we focus on Fourier space). They proved in the paper that if N_t and N_ω are the number of non-zero elements of a length N signal in the time and Fourier space respectively. Then

$$N_t N_\omega \geq N$$

It tells us that Fourier sparse functions must be dense in time space, which means that we can get meaningful (non-zero) samples easily. Also, this discrete-time principle shows that a wide-band signal can be reconstructed from narrow-band data, which suggests that if the signal is sparse in Fourier space, then by carefully designing the sampling scheme, the number of total samples needed to recover the signal can be much less than the bandwidth N.

Because of the excellent properties (sub-sampling and sub-linear runtime), some of the existing SFT techniques that can work with discrete data have been applied to many signal processing problems including, e.g., GPS signal acquisition [11], analog-to-digital conversion [13, 15], and wideband communication/spectrum sensing [14, 12]. However, there also exist some other SFT algorithms [4, 6] which have good runtime and error guarantee. They have not been widely used because they assume that the algorithm can sample anywhere they want, which is not a realistic assumption in many applications. Actually, in most of the real-world problems, one can only expect to have equally spaced samples. In this thesis, we propose a generic method of transforming any noise robust SFT algorithm into a sublinear-time sparse DFT algorithm which rapidly approximates $\hat{\mathbf{f}}$ from a given input vector $\mathbf{f} \in \mathbb{C}^N$. As a result, we can construct several new sublinear-time sparse DFT algorithms from existing sparse Fourier algorithms which utilize unequally spaced function samples [8, 2, 4, 6, 10]. Moreover, for the new discrete SFT algorithm based on the Algorithm 3 in [4], we show that it can always identify a vector $\mathbf{v}$ which has only s non-zero elements and satisfies

$$\| \hat{\mathbf{f}} - \mathbf{v} \|_2 \leq \| \hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{opt} \|_2 + \frac{33}{\sqrt{s}} \| \hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{opt} \|_1 + 198\sqrt{s} \| f \|_\infty N^{-r} \tag{1.2}$$

where $1 \leq r \leq \frac{N}{36}$ with runtime $poly(s, r, \log N)$. The $\hat{\mathbf{f}}_s^{opt}$ indicates the optimal s-term approximation of $\hat{\mathbf{f}}$. At the very end, we show empirically that the best of our new algorithms can outperform both FFTW[24] and sFFT2.0[23] in the sense of runtime and robustness when the vector length N is large.

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

$$f(x) = \sum_{j=1}^{n} \sum_{\omega \in S_j} c_\omega e^{\mathbb{i}\omega x} \tag{1.3}$$

where each unknown S_j has simplifying structure. (e.g., has $|x - y| < B \ll N$ for all $x, y \in S_j$). Instead of using FFT, which takes $\mathcal{O}(N \log N)$ time, to solve this problem, we consider a faster deterministic Sparse Fourier Transform (SFT) methods which are guaranteed to recover such f using a number of samples and operations that scale at most polynomially in both $\sum_{j=1}^{n}|S_j|$ and $\log(N)$ [30].

1.2 Related work

The first work that tried to solve the sparse approximate DFT problem can be found in [47], in which they designed an algorithm based on the Hadamard Transform, i.e. the Fourier transform over the Boolean cube. Later, a polynomial time algorithm to interpolate a sparse polynomial was developed in [49]. The method in this paper inspired the authors of [40], in which they described an algorithm that can be used to approximate the DFT when N is a power of 2. In the early 2000s, people paid a lot of attention to the sparse approximation problem in Fourier space. The first algorithm with sub-linear runtime and the sub-sampling property was given in [39], in which they give a randomized algorithm for finding an near-optimal s-term approximation $\widehat{\mathbf{y}}$, with probability $1-\delta$, in $\text{poly}(s, \log N, \log(1/\delta), 1/\epsilon)$ time. Here given the input signal $\mathbf{f}$, and let $\widehat{\mathbf{f}}_s^{opt}$ be the optimal s-term approximation to $\widehat{\mathbf{f}}$. The near-optimal s-term approximation $\widehat{\mathbf{y}}$ is defined as

$$\| \widehat{\mathbf{f}} - \widehat{\mathbf{y}} \|_2^2 \leq (1+\epsilon) \| \widehat{\mathbf{f}} - \widehat{\mathbf{f}}_s^{opt} \|_2^2 \tag{1.4}$$

Equation (1.4) is known as an ℓ_2/ℓ_2 guarantee, which is a strong guarantee and it was shown in [37] that this guarantee cannot hold for a sub-linear deterministic algorithm. One thing worth to mention here is that the runtime of the algorithm in [39] is quadratic in sparsity s. This algorithm was modified later in [40] to reduce the power of s in runtime to 1. However, unlike [39] which uses only equally spaced samples, the algorithm in [40] are allowed to use unequally spaced data. Since the standard FFT can only be used for equally-spaced data, the algorithm in [38] was applied to reduce the runtime when calculating the DFT for unequally spaced samples. Even so, the algorithm can only outperform the FFT for the signal with small s. Shortly after that, as an extension of the algorithm in [46], [35] proved hard-core predicates using list decoding. However, the runtime of the algorithm in [35] has a high dependence on sparsity compared with [39] and [40].

All the SFT algorithms above are randomized algorithms. This means they have small probability to fail to give the correct or optimal recovery on each input signal. Thus, they are not appropriate for long-lived failure intolerant applications. The first deterministic sublinear time SFT algorithm was developed in [33] based on the deterministic Compressed Sensing results of Cormode and Muthukrishnan (CM)[25, 51, 50]. A simpler optimized version of this algorithm was given in [2], which has similar runtime/sampling bounds ($\mathcal{O}(s^2 \log^4 N)$) to the one in [40]. Later, a further modified SFT algorithm was provided in [4]. It showed simple methods for extending the improved sparse Fourier transforms to higher dimensional settings. More specifically, the algorithm can find the near optimal s-term approximation for any given input function, $f : [0, 2\pi]^D \to \mathbb{C}$ in $\mathcal{O}(s^2 D^4)$ time (neglecting logarithmic factors). The algorithms in [33, 2, 4] are all aliasing-based search algorithms [9], which means they rely on the combinatorial properties of aliasing among frequencies in the sub-sampled DFTs. The algorithms first find the residues of the significant frequencies modulo with different sample lengths, and then use the Chinese Reminder Theorem to finish the reconstruction. In this series of work, the error bound is of the form

$$\| \widehat{\mathbf{f}} - \widehat{\mathbf{y}} \|_2 \leq \| \widehat{\mathbf{f}} - \widehat{\mathbf{f}}_s^{opt} \|_2 + \frac{1}{\sqrt{s}} \| \widehat{\mathbf{f}} - \widehat{\mathbf{f}}_s^{opt} \|_1 \qquad (1.5)$$

In 2012, an algorithm that can compute the non-zero coefficients in time $\mathcal{O}(s \log N)$ was given in [23]. The algorithm leverages techniques from digital signal processing (notably Gaussian and Dolph-Chebyshev filters) and unlike other randomized algorithms, this algorithm is not iterative. It identifies and estimates the s largest coefficients in one shot. This makes the algorithm have good performance even when the sparsity is large. The algorithm has runtime $\mathcal{O}(\log N \sqrt{N s \log N})$ and it satisfy the so-called $\ell_\infty / \ \ell_2$ guarantee. Specifically, for a precision parameter $\delta = 1/N^{\mathcal{O}(1)}$ and a constant $\epsilon > 0$, the algorithm outputs $\widehat{\mathbf{y}}$ such that:

$$\| \widehat{\mathbf{f}} - \widehat{\mathbf{y}} \|_\infty^2 \leq \frac{\epsilon}{s} \| \widehat{\mathbf{f}} - \widehat{\mathbf{f}}_s^{opt} \|_2 + \delta \| \mathbf{f} \|_1^2 \tag{1.6}$$

with probability $1 - 1/N$.

Around the same time, a new deterministic SFT algorithm based on phase encoding was presented in [10] with runtime $\mathcal{O}(s \log s)$. The authors first gave a few observations relating the Fourier coefficients of time-shifted samples to unshifted samples of the input function. Then they used this observation to detect when aliasing between two or more frequencies has occurred, as well as to determine the value of unaliased frequencies. The algorithm has a simple structure and is easy to implement, however, it can only be used for noiseless signals and hence cannot be utilized in real-world applications due to imprecise instrumentation or noisy environment. Later, the authors extended their algorithm to noisy setting by using the multiscale error-correcting method [6]. This new algorithm is an adaptive algorithm with runtime $\mathcal{O}(s \log(s) \log(N/s))$. Most recently, another phase encoding SFT algorithm was presented in [54], in which the authors developed an efficient algorithm for sparse FFT for higher dimensional signals by extending some of the ideas in [10]. The algorithm was using the so-called “partial unwrapping” and “tilting methods”. These two methods allow the algorithm in [54] to efficiently compute the sparse FFT of higher dimensional signals with $\mathcal{O}(Ds \log s)$ computational complexity and $\mathcal{O}(Ds)$ sampling size for any given input function $f : [0, 2\pi]^D \to \mathbb{C}$. There are also some researches try to conquer the SFT problem in Fourier space from other aspects. For example, from implementation [28, 29], and hardware [31, 32] perspectives.

1.3 Thesis outline

The remainder of this thesis is organized as follows. In Chapter 2 we set up the notations that will be used throughout the thesis. In Chapter 3 we introduce the background of sparse Fourier transform and then focus on two specific SFT algorithms: GFFT (stands for **G**opher **F**ast **F**ourier **T**ransform) [33, 2, 4, 8] and CLW-SFT (stands for **C**hristlieb **L**awlor **W**ang - **S**parse **F**ourier **T**ransform) [6, 10]. For both of them we first illustrate the basic idea of the SFT algorithm with the simplest single frequency case ($s = 1$), and then we talk about how these two SFT algorithms work in general case ($s \geq 2$). In Chapter 4 we talk about our approach to building fully discrete SFT algorithm based on periodic Gaussian function and fast discrete convolution. We show that by using our approach, the evaluation of the value of convolution function $(g * f)$ at the desired point can always be approximated very accurately using only the given discrete data with runtime $\mathcal{O}(\log N)$. Then in Chapter 5, we apply our approach to GFFT. We first show that the GFFT is noise robust by extending the Theorem 7 in [4]. Then we give the runtime and error guarantee for the fully discrete SFT algorithm based on GFFT in both deterministic and randomized version. In Chapter 6 we empirically evaluate the performance of DMSFT (generated from GFFT) and CLW-DSFT (generated from CLW-SFT), and then we compare their runtime and robustness characteristics with FFTW 3.3.4[1] and sFFT 2.0[2]. Finally, in Chapter 7 we present a deterministic sparse Fourier transform algorithm which breaks the quadratic-in-sparsity runtime bottleneck for a large class of periodic functions exhibiting structured frequency support. We focus on the numerical experiments of the structured SFT algorithm and point the reader to [30] for more details of the theoretical guarantee.

[1]http://www.fftw.org/

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

Chapter 2

Notation and Setup

In this chapter we define the notations that will be used later. The Fourier series representation of a $2\pi-$periodic function $f : [-\pi, \pi] \to \mathbb{C}$ will be denoted by

$$f(x) = \sum_{\omega \in \mathbb{Z}} \widehat{f}_\omega e^{i\omega x}$$

with its Fourier coefficients given by

$$\widehat{f}_\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-i\omega x} \, dx.$$

We let $\widehat{f} := \left\{ \widehat{f}_\omega \right\}_{\omega \in \mathbb{Z}}$ represent the infinite sequence of all Fourier coefficients of f. Given two $2\pi-$periodic functions f and g we define the convolution of f and g at $x \in \mathbb{R}$ to be

$$(f * g)(x) \;=\; (g * f)(x) \;:=\; \frac{1}{2\pi} \int_{-\pi}^{\pi} g(x - y) f(y) \, dy.$$

This definition, coupled with the definition of the Fourier transform, yields the well-known equality

$$\widehat{f * g}_\omega = \widehat{f}_\omega \widehat{g}_\omega \;\forall \omega \in \mathbb{Z}.$$

We may also write $\widehat{f * g} = \widehat{f} \circ \widehat{g}$ where $\circ$ denotes the Hadamard product.

For any $N \in \mathbb{N}$, define the Discrete Fourier Transform (DFT) matrix $F \in \mathbb{C}^{N \times N}$ by

$$F_{\omega,j} := \frac{(-1)^{\omega}}{N} \mathbb{e}^{-\frac{2\pi \mathbb{i} \cdot \omega \cdot j}{N}},$$

and let $B := \left(-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right] \cap \mathbb{Z}$ be a set of N integer frequencies centered at 0. Furthermore, let $\mathbf{f} \in \mathbb{C}^N$ denote the vector of equally spaced samples from f whose entries are given by

$$f_j := f\left(-\pi + \frac{2\pi j}{N}\right)$$

for $j = 0, \ldots, N-1$. One can now see that if

$$f(x) = \sum_{\omega \in B} \widehat{f}_{\omega} \mathbb{e}^{\mathbb{i}\omega x},$$

then

$$F\mathbf{f} = \hat{\mathbf{f}} \tag{2.1}$$

where $\hat{\mathbf{f}} \in \mathbb{C}^N$ denotes the subset of $\widehat{f}$ with indices in B, and in vector form. More generally, bolded lower case letters will always represent vectors in $\mathbb{C}^N$ below.

For any positive integer p, let $\mathbf{f}_{[p]}$ denotes the length p vector whose elements are equally spaced samples from f

$$\mathbf{f}_{[p]}(j) := f\left(-\pi + \frac{2\pi j}{p}\right)$$

for $j = 0, ..., p-1$. Let $\mathbf{f}_{[p,\epsilon]}$ be the discrete array by sampling $f(x)$ at rate $\frac{1}{p}$ starting at $x = \epsilon$. That is, for $j = 0, ..., p-1$

$$\mathbf{f}_{[p,\epsilon]}(j) = f\left(-\pi + 2\pi(\frac{j}{p} + \epsilon)\right)$$

Define $\mathbf{f}^n_{[p]}$ and $\mathbf{f}^n_{[p,\epsilon]}$ to be the noisy versions of vectors $\mathbf{f}_{[p]}$ and $\mathbf{f}_{[p,\epsilon]}$, that is

$$\mathbf{f}^n_{[p]} = \mathbf{f}_{[p]} + \mathbf{n} \tag{2.2}$$

$$\mathbf{f}^n_{[p,\epsilon]} = \mathbf{f}_{[p,\epsilon]} + \mathbf{n} \tag{2.3}$$

where $\mathbf{n}$ is a length p vector whose entries are complex i.i.d standard normal random variables, i.e. $\mathbf{n}_j = \sigma(\eta^1_j + \mathbb{i}\eta^2_j)$, where σ is the standard deviation of the noise. Finally, let $\hat{\mathbf{f}}_{[p]}$, $\hat{\mathbf{f}}_{[p,\epsilon]}$, $\hat{\mathbf{f}}^n_{[p]}$ and $\hat{\mathbf{f}}^n_{[p,\epsilon]}$ be the discrete Fourier transform vector of $\mathbf{f}_{[p]}$, $\mathbf{f}_{[p,\epsilon]}$, $\mathbf{f}^n_{[p]}$ and $\mathbf{f}^n_{[p,\epsilon]}$ respectively.

As mentioned above, $\widehat{f} := \left\{\widehat{f}_\omega\right\}_{\omega\in\mathbb{Z}}$ is the infinite sequence of all Fourier coefficients of f. For any subset $S \subseteq \mathbb{Z}$ we let $\widehat{f}|_S \in \mathbb{C}^{\mathbb{Z}}$ be the sequence $\widehat{f}$ restricted to the subset S, so that $\widehat{f}|_S$ has terms $\left(\widehat{f}|_S\right)_\omega = \widehat{f}_\omega$ for all $\omega \in S$, and $\left(\widehat{f}|_S\right)_\omega = 0$ for all $\omega \in S^c := \mathbb{Z}\setminus S$. Note that $\hat{\mathbf{f}}$ above is exactly $\widehat{f}|_B$ excluding its zero terms for all $\omega \notin B$. Thus, given any subset $S \subseteq B$, we let $\hat{\mathbf{f}}|_S \in \mathbb{C}^N$ be the vector $\hat{\mathbf{f}}$ restricted to the set S in an analogous fashion. That is, for $S \subseteq B$ we will have $\left(\hat{\mathbf{f}}|_S\right)_\omega = \hat{\mathbf{f}}_\omega$ for all $\omega \in S$, and $\left(\hat{\mathbf{f}}|_S\right)_\omega = 0$ for all $\omega \in B \setminus S$.

Given the sequence $\widehat{f} \in \mathbb{C}^{\mathbb{Z}}$ and $s \leq N$, we denote by $R^{\text{opt}}_s\left(\widehat{f}\right)$ a subset of B containing s of the most energetic frequencies of f; that is

$$R^{\text{opt}}_s\left(\widehat{f}\right) := \{\omega_1, \dots, \omega_s\} \subseteq B \subset \mathbb{Z}$$

where the frequencies $\omega_j \in B$ are ordered such that

$$\left|\widehat{f}_{\omega_1}\right| \geq \left|\widehat{f}_{\omega_2}\right| \geq \cdots \geq \left|\widehat{f}_{\omega_s}\right| \geq \cdots \geq \left|\widehat{f}_{\omega_N}\right|.$$

Here, if desired, one may break ties by also requiring, e.g., that $\omega_j < \omega_k$ for all $j < k$ with

$\left|\widehat{f}_{\omega_j}\right| = \left|\widehat{f}_{\omega_k}\right|$. We will then define $f_s^{\mathrm{opt}} : [-\pi, \pi] \to \mathbb{C}$ based on $R_s^{\mathrm{opt}}\left(\widehat{f}\right)$ by

$$f_s^{\mathrm{opt}}(x) := \sum_{\omega \in R_s^{\mathrm{opt}}\left(\widehat{f}\right)} \widehat{f}_\omega \mathbb{e}^{\mathbb{i}\omega x}.$$

Any such 2π-periodic function f_s^{opt} will be referred to as an optimal s-term approximation to f. Similarly, we also define both $\widehat{f}_s^{\mathrm{opt}} \in \mathbb{C}^{\mathbb{Z}}$ and $\hat{\mathbf{f}}_s^{\mathrm{opt}} \in \mathbb{C}^N$ to be $\widehat{f}|_{R_s^{\mathrm{opt}}\left(\widehat{f}\right)}$ and $\hat{\mathbf{f}}|_{R_s^{\mathrm{opt}}\left(\widehat{f}\right)}$, respectively.

Chapter 3

Background and Review

In this chapter, we introduce the background of sparse Fourier transform and then focus on two SFT algorithms: the CLW-SFT developed by Andrew Christlieb, David Lawlor and Yang Wang in 2013 [6]; and GFFT developed by Mark Iwen in 2013 [4].

The primary challenge of sparse Fourier transform algorithm is how to isolate one frequency from other frequencies efficiently and accurately. It can be shown that if the signal contains only one frequency and it is noiseless, then we can find this single frequency and its corresponding coefficient by using only two samples from the signal. However, if noise contaminates the signal, then one will need a logarithmic number of samples to guarantee to recover the frequency. Since all the methods for general cases can be solved by reducing it to several sub-problems involving only one frequency, in Section 3.1.1 and 3.2.1, we discuss two different approaches to solving this simple single frequency problem. This reducing process can be achieved by grouping subsets of Fourier space together into a small number of bins. Then we can recover the isolated frequency by using the techniques mentioned earlier. This means the runtime of a (sparse) Fourier Transform algorithm is closely related to the number of bins. One of the ideas about how to set up the bins is to let each bin corresponding to a subset of bandwidth. Usually, this approach works well when the significant frequencies are distributed uniformly in the bandwidth (the distance between different significant frequencies in Fourier space is relatively far). However, this method can fail when two non-zero frequencies are very close to each other. In the worst case, one will need N bins to guarantee the isolation; this leads to totally $O(N \log N)$ runtime, which is equivalent to the runtime of Fast Fourier Transform(FFT).

On the other hand, one can also design or set up the bins cleverly to significantly reduce the number of bins. However, since each of the significant frequencies have to be isolated at least once, one will need at least s bins to guarantee the correctness of recovery, where s is the number of significant frequencies (sparsity) of the signal in Fourier space.

The remainder of this chapter is organized as follows. In Section 3.1 we introduce the sparse Fourier algorithm based on the phase encoding method[10][6], in Section 3.2 we briefly review another method developed by Mark Iwen, which is an aliasing-based search method. For both Section 3.1 and 3.2, we start with the simplest single frequency case, and then we move to the more general case of which the signal contains more than one significant frequency.

3.1 CLW-SFT

CLW-SFT is a sparse Fourier transform algorithm developed by Andrew Christlieb, David Lawlor, and Yang Wang[6]. It is a phase encoding method. The noiseless version of this algorithm was first published in [10] in 2013. It is an adaptive algorithm which has running time $O(s \log s)$. Although fast, it is not robust to noise. Unfortunately, any signal recorded from real-world data is contaminated due to either imprecise instrumentation or noisy environment. Later in another paper [6], they developed a multi-scale sub-linear Fourier algorithm with runtime $O(s^2 \log(s))$. It is worth to mention that this multi-scale algorithm is robust to high-level noise when measuring the error under Earth Mover Distance. For more details, we point the reader to [6] and [10].

Next, we start with the simplest single frequency case, and then we talk about how the algorithm works in the general case.

3.1.1 Single Frequency Recovery

Let us assume that the signal is noiseless and it contains only one significant frequency ω with corresponding coefficient $\widehat{\mathbf{f}}_\omega$. Then for an integer p and $\epsilon \leq \frac{1}{N}$ (get ride of wrap-around aliasing), we take two sets of samples $\mathbf{f}_{[p]}$ and $\mathbf{f}_{[p,\epsilon]}$ and then take the discrete Fourier transform. We have:

$$\widehat{\mathbf{f}}_{[p]}(h) = \begin{cases} p\widehat{\mathbf{f}}_\omega, & h = \omega \pmod p \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

and

$$\widehat{\mathbf{f}}_{[p,\epsilon]}(h) = \begin{cases} p\widehat{\mathbf{f}}_\omega e^{2\pi \mathbb{i} \epsilon \omega}, & h = \omega \pmod p \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

for $h = 0, ..., p-1$.

Now we can easily recover the significant frequency ω by using the equation below:

$$\omega = \frac{1}{2\pi\epsilon} \text{Arg}\left(\frac{\widehat{\mathbf{f}}_{[p,\epsilon]}(h)}{\widehat{\mathbf{f}}_{[p]}(h)}\right) \tag{3.3}$$

where Arg(z) denote the phase angle of the complex number z in $(-\pi, \pi]$. It is clear that as long as $\epsilon \leq \frac{1}{N}$, the number of samples we need to recover the significant frequency is $\mathcal{O}(1)$ (independent to the bandwidth N).

3.1.2 General Case

From above we know that if the signal is noiseless and contains only one frequency, then we can find the significant frequency in $\mathcal{O}(1)$ time. In this section, we talk about the algorithm in [6] in the more general case.

Assume now that there are $s > 1$ significant frequencies $\{\omega_j\}_{j=1}^{s}$ in the signal function and assume their corresponding coefficients are $\{\widehat{\mathbf{f}}_{\omega_j}\}_{j=1}^{s}$. Then the method in section 3.1.1 can no longer guarantee to find the significant frequencies all the time. This is because when aliasing occurs the equation 3.1 and 3.2 is not always true and so the reconstruction via 3.3 is no longer valid. More specifically, when there are more than one significant frequencies in the signal, if we still take two sets of samples $\mathbf{f}_{[p]}$ and $\mathbf{f}_{[p,\epsilon]}$, then 3.1 and 3.2 becomes

$$\widehat{\mathbf{f}}_{[p]}(h) = p \sum_{\omega_l \equiv h \pmod{p}} \widehat{\mathbf{f}}_{\omega_l} \tag{3.4}$$

and

$$\widehat{\mathbf{f}}_{[p]}(h) = p \sum_{\omega_l \equiv h \pmod{p}} \widehat{\mathbf{f}}_{\omega_l} e^{2\pi \mathbb{i} \omega_l \epsilon} \tag{3.5}$$

for $h = 0, ..., p-1$. Now it is clear that we cannot use 3.3 to reconstruct the frequency when two or more frequencies are congruent modulo the sampling rate p. However, if we can detect when the aliasing happens, we can always change the sampling rate p to some other number to avoid the aliasing. The Lemma below in [10] helps to solve the aliasing detection problem.

Lemma 1. *Let $p > 1$ and $h \in \{0, 1, ..., p-1\}$. Then for almost all $\epsilon > 0$ we have $|\widehat{\mathbf{f}}_{[p,\epsilon]}(h)| \neq |\widehat{\mathbf{f}}_{[p]}(h)|$.*

In practice, to detect the aliasing, we only need to set a tolerance τ in order to accept or reject frequencies according to the criterion

$$\left| \frac{\widehat{\mathbf{f}}_{[p,\epsilon]}(h)}{\widehat{\mathbf{f}}_{[p]}(h)} - 1 \right| \leq \tau \tag{3.6}$$

The above inequality allows us to detect whether or not two or more frequencies are aliased, so that we only add non-aliased terms to our representation. Then we quickly discover all frequencies present in the signal function f by choosing a different value for p in a subsequent iteration. Please see [10] for a more rigorous prove.

The previous phase shift and aliasing detection method can only be used for the noiseless signal because the shift ϵ is sensitive to noise. However, due to imprecise instrumentation or noisy environments, any signal recorded from the real-world will more or less contain some errors. Next, we introduce the method in [6] to show that it is possible to get a noise robust method by modifying the phase shift and aliasing detection method mentioned above.

Now, assume the two sets of samples we have are $\mathbf{f}^n_{[p]}$ and $\mathbf{f}^n_{[p,\epsilon]}$. Notice that the super scribe n means the signal is contaminated by complex Gaussian noise with standard deviation σ. It has been proved in [6] that if $\widehat{\mathbf{f}}^n_{[p]}$ and $\widehat{\mathbf{f}}^n_{[p,\epsilon]}$ denote the DFT of $\mathbf{f}^n_{[p]}$ and $\mathbf{f}^n_{[p,\epsilon]}$, then for $h \in \{0, 1, ..., p-1\}$ we have

$$\left| \mathrm{Arg}\left(\frac{\widehat{\mathbf{f}}^n_{[p,\epsilon]}(h)}{\widehat{\mathbf{f}}^n_{[p]}(h)} \right) - 2\pi\omega\epsilon \right| \leq \mathcal{O}\left(\frac{\sigma\sqrt{p}}{|\widehat{\mathbf{f}}_{[p]}(h)|}\right) \tag{3.7}$$

This means if we use the method stated previously, then the noise frequency estimate ω satisfy

$$|\tilde{\omega} - \omega| \leq \mathcal{O}\left(\frac{\sigma\sqrt{p}}{2\pi\epsilon|\widehat{\mathbf{f}}_{[p]}(h)|}\right) \tag{3.8}$$

where ω is the true frequency. Turns out that if the noise level is low, then we can fix this problem with a minor modification. The approach is simply round the noisy frequency estimate $\tilde{\omega}$ to the nearest integer of the form $np + h$. The improved estimate $\tilde{\omega}$' is therefore given by

$$\tilde{\omega}' = p \cdot \text{round}(\frac{\tilde{\omega} - h}{p}) + h \tag{3.9}$$

where round(x) returns the neatest integer to x. This modification works because when the noise level is low the noise frequency estimate $\tilde{\omega}$ is deviated by less than $p/2$ from the true frequency ω. However, this is not true for large noise level.

The multi-scale method in [6] solves this problem by using multiple shifts ϵ_q. The key idea of the multi-scale algorithm is that if we know $|\omega| < \frac{L}{2}$ for some positive number L, then it is enough to let $\epsilon < \frac{1}{L}$ to identify ω. Later we will see that this fact allows us to use much coarser sampling rate than the previous simple rounding method 3.9.

The algorithm starts by choosing an appropriate p and $\epsilon < \frac{1}{2N}$ and calculating the initial frequency estimates $\tilde{\omega}_j^0$, $j = 1, ..., s^*$ with

$$\tilde{\omega}_j^0 = p \cdot \text{round}\left(\frac{1}{2\pi\epsilon p}\left(\text{Arg}\left(\frac{\widehat{\mathbf{f}}^n_{[p,\epsilon]}(h)}{\widehat{\mathbf{f}}^n_{[p]}(h)}\right) - h\right)\right) + h \tag{3.10}$$

which will deviate from the true frequency ω_j by approximately $\sigma/2\pi\epsilon\sqrt{p}$. We then use an iterative frequency estimation procedure to correct for the error in phase. For each of the s^* estimated frequencies, we increase the ϵ with a factor of 2^B, where B is a parameter. Then we take several other sets of samples

$$\begin{aligned} \mathbf{f}^n_{[p]}(q) &= e^{-2\pi \mathbb{i} \tilde{\omega}_j^0 q/p} f(\frac{q}{p}) && (3.11) \\ \mathbf{f}^n_{[p,\epsilon]}(q) &= e^{-2\pi \mathbb{i} \tilde{\omega}_j^0 (q/p + \epsilon 2^B)} f(\frac{q}{p} + \epsilon 2^B) && (3.12) \end{aligned}$$

for $j = 1, ..., s^*$, and take DFTs on them. Note now that we can improve our estimation from 3.10 by letting

$$\tilde{\omega}_j^1 = \tilde{\omega}_j^0 + p \cdot \text{round}\left(\frac{1}{2\pi\epsilon 2^B p} \text{Arg}\left(\frac{\widehat{\mathbf{f}}^n_{[p,\epsilon]}(0)}{\widehat{\mathbf{f}}^n_{[p]}(0)} \right) \right) \tag{3.13}$$

This is because the way we collect samples in 3.11 and 3.12 will shift the peak location $h = \omega_j(\text{ mod } p)$ down by $\tilde{\omega}_j^0(\text{ mod } p)$. So we expect to see the peak corresponding to ω_j to appear at the zero frequency in the demodulated signal. Also we notice that the difference between thezero frequency of the shifted and unshifted DFT values is proportional to $\tilde{\omega}_j^0 - \omega_j$, which lead us to 3.13. Then we just repeat this process until the algorithm converge. For the coefficients, we simply take the median of the real and imaginary parts of the estimates. Note that the correction in 3.13 make the deviation of the new estimation smaller by a factor of 2^B. In another words, for each iteration, we improve the accuracy of the estimation by a digit in the binary sense. The multi-scale algorithm is robust to high level noise and it has averaged-case runtime $\mathcal{O}(s^2 \log(s) \log(N))$ and sampling complexity $\mathcal{O}(s^2 \log(N))$. For more details about the multi-scale SFT algorithm, we encourage the readers to see [6].

3.2 GFFT

The GFFT is a deterministic SFT algorithm built by Mark Iwen in 2010 [2]. The algorithm was first implemented by Mark Iwen and Ben Segal in 2013 [8]. In this section, we first illustrate the idea of the GFFT algorithm by accomplishing the fundamental single frequency recovery task; then we introduce the algorithm in the more general setting.

3.2.1 Single Frequency Recovery

Let us assume in this section that there is only one significant frequency ω in the signal function f.

The GFFT is an aliasing-based search method. Its idea works particularly well when the bandwidth N is a product of several smaller relatively prime numbers. For example, assume the bandwidth $N = p_1 \cdot p_2 \cdot p_3$, where p_1, p_2 and p_3 are co-prime numbers. Now, in order to identify the significant frequency, we take three sets of equally spaced samples with length p_1, p_2 and p_3, and call them $\mathbf{f}_{p_1}$, $\mathbf{f}_{p_2}$ and $\mathbf{f}_{p_3}$ respectively. Let $\widehat{\mathbf{f}}_{p_1}$, $\widehat{\mathbf{f}}_{p_2}$ and $\widehat{\mathbf{f}}_{p_3}$ be the DFTs of $\mathbf{f}_{p_1}$, $\mathbf{f}_{p_2}$ and $\mathbf{f}_{p_3}$. Notice that $\widehat{\mathbf{f}}_{p_1}$ has exactly one non-zero element since there is only one significant frequency in the signal. If we assume the indexes of this non-zero element is r_1. Then we know that

$$\omega \equiv r_1 \bmod p_1 \tag{3.14}$$

Similarly, from $\widehat{\mathbf{f}}_{p_2}$ and $\widehat{\mathbf{f}}_{p_3}$ we know

$$\omega \equiv r_2 \bmod p_2 \tag{3.15}$$

$$\omega \equiv r_3 \bmod p_3 \tag{3.16}$$

Once we have collected these modulo we can reconstruct ω via Chinese Remainder Theorem (CRT).

Theorem 1. ***Chinese Remainder Theorem**: An integer x is uniquely specified modulo N by its remainders modulo m relatively prime integers $p_1, ..., p_m$ as long as $\Pi_{l=1}^{m} p_l \geq N$.*

Now it is clear that from the CRT, ω can be uniquely determined by 3.14, 3.15 and 3.16 since we have $N = p_1 \cdot p_2 \cdot p_3$. Notice that above method always work as long as $N \leq p_1 \cdot p_2 \cdot p_3$.

We borrow the example in [2] to show that this method can dramatically decrease the needed samples to find the significant frequency. Assume $N = 10^6$ and we have three FFTs with length 100, 101 and 103 samples to determine that $\omega \equiv 34 \bmod 100$, $\omega \equiv 3 \bmod 101$ and $\omega \equiv 1 \bmod 103$, respectively. Using $\omega \equiv 34 \bmod 100$ and $\omega \equiv 3 \bmod 101$, we can see that $\omega = 3134 \bmod 100 \cdot 101$. This comes from the fact that if

$$x \equiv a_1 \bmod n_1 \tag{3.17}$$

$$x \equiv a_2 \bmod n_2 \tag{3.18}$$

Then we have $x = a_1 m_2 n_2 + a_2 m_1 n_1$, where m_1 and m_2 are the integers which satisfy

$$m_1 n_1 + m_2 n_2 = 1 \tag{3.19}$$

Notice that 3.19 is from the Bezout's identity and can be solved by using extended Euclidean algorithm. By similar work, we can use $\omega = 3134 \bmod 100 \cdot 101$ and $\omega \equiv 1 \bmod 103$ to solve for ω. This gives us $\omega = 104134$ by Chinese Remainder Theorem.

From above example, we can see that the number of the samples we have used to determine ω is significantly less than N: the conventional DFT method requires 10^6 samples, however, by using the CRT, we needed only $100+101+103 = 304$ samples from f. Moreover, this method is deterministic, and hence it has no chance to fail. In next section, we show that we can also use the CRT to build a deterministic SFT algorithm if we deal with the potential difficulties caused by frequency collisions carefully.

3.2.2 General Case

Let us start with two frequency case. In this scenario, the signal function f is in the form

$$f(x) = C_1 e^{\mathbb{i}\omega_1 x} + C_2 e^{\mathbb{i}\omega_2 x} \tag{3.20}$$

If we assume that we know some integer n such that

$$\omega_1 \bmod n \neq \omega_2 \bmod n \tag{3.21}$$

Then we say that n separates ω_1 and ω_2 and we can take advantage of this information to find ω_1 and ω_2 as follows. Assume $h_1 = \omega_1 \bmod n$ and $h_2 = \omega_2 \bmod n$. Here we call h_1 and h_2 residues modulo n. Notice that h_1 and h_2 are simply the index of the two largest entries of $\widehat{\mathbf{f}}_{[n]}$. Now if we take an other set of samples $\mathbf{f}_{[c \cdot n]}$, where c is relatively prime to n, and then take DFT to it. Then the elements $\widehat{\mathbf{f}}_{[c \cdot n]}$ will all be 0 except those satisfy

$$\begin{aligned} \widehat{\mathbf{f}}_{[c \cdot n]}(\omega_1 \bmod cn) &= C_1 \neq 0 & (3.22) \\ \widehat{\mathbf{f}}_{[c \cdot n]}(\omega_2 \bmod cn) &= C_2 \neq 0 & (3.23) \end{aligned}$$

However, before we proceed, we must fix two problems: first, the above procedure will not give us the residues if ω_1 and ω_2 modulo relatively prime numbers because n divides cn; second, if $C_1 = C_2$ then we will not be able to tell which residues modulo each $cn-$ value corresponding to ω_1 versus ω_2. The author of [2] uses the method inspired by [51], [50] and [52] to solve these difficulties. We use an example to better explain the approach in [2].

Assume we have both $\widehat{\mathbf{f}}_{[n]}$ and $\widehat{\mathbf{f}}_{[2n]}$. Now let us focusing on ω_1. Since we have $\omega_1 \bmod n =$

h_1, so for $2n$ we must have one of the two equations below

$$\omega_1 \bmod 2n = h_1, \text{ or } \omega_1 \bmod 2n = h_1 + n \tag{3.24}$$

By using the above fact we can easily determine $\omega_1 \bmod 2n$ by

$$\omega \bmod 2n = \begin{cases} h_1, & \text{if } \left|\widehat{\mathbf{f}}_{[n]}(h_1) - \widehat{\mathbf{f}}_{[2n]}(h_1)\right| < \left|\widehat{\mathbf{f}}_{[n]}(h_1) - \widehat{\mathbf{f}}_{[2n]}(h_1+n)\right| \\ h_1 + n, & \text{otherwise} \end{cases} \tag{3.25}$$

Now, once we know $\omega_1 \bmod 2n$ we can calculate $\omega_1 \bmod 2$ by

$$\omega_1 \bmod 2 = (\omega_1 \bmod 2n) \bmod 2. \tag{3.26}$$

If 2 is relatively prime to n, then we can use both residues in the CRT. By setting c to be several co-prime numbers, i.e. $c = 2, 3, 5, 7, ...\mathcal{O}(\log N)$, we can reconstruct ω_1. The ω_2 can be reconstructed in a similar way.

The above method only works under the assumption that we know an integer n separates ω_1 and ω_2. This assumption is not realistic because usually, we do not have any prior knowledge concerning separating $n-$values. Next, we will utilize an example to show that we can still recover the significant frequencies even if we do not have any knowledge about the n values. This method was inspired by similar number theoretic group testing strategies used in [53] and [55].

Assume $N = 10^6$ and let n_j, $j = 1, 2, 3, 4, 5$ be 5 relatively prime numbers 100, 101, 103, 107 and 109 respectively. Now we can observe that the product of any three of these co-prime numbers is greater than N. Combine this fact with the CRT we know that as long as $\omega_1 \neq \omega_2$, they can collide modulo at most two of the relatively prime n_j values. This is because if they collide at three n_j values, then they must be identical based on the CRT. This fact gives us a way to identify ω_1 and ω_2. We can apply the previous recovery method for each of the five n_j values. Because each time when we apply the method with a different n_j value, we will get at most two unique frequencies, so at the very end we will have at most 10 answers (with multiplicity). Since we know that the majority of our n_j values (3 out of 5) must separate ω_1 and ω_2, so ω_1 and ω_2 must appear in our answers at least 3 times each. Also from the CRT we know that the incorrect frequency can appears in our answers at most twice. So we only need to return the answers which appears at least three times, that will give us ω_1 and ω_2. We can use the similar majority theory to find the coefficient of each frequency. This method also works for signals with sparsity greater than 2. Actually, it has been proved in [33] that it is enough to use $3s\lfloor \log_s N \rfloor + 1$ co-prime numbers to guarantee to find the correct set of significant frequencies. Moreover, we will obtain a $s-$term representation $\widehat{\mathbf{f}}|_S$ of the signal function f in Fourier space which satisfies

$$\| \widehat{\mathbf{f}} - \widehat{\mathbf{f}}|_S \|_2 < \| \widehat{\mathbf{f}} - R_s^{opt}(\widehat{f}) \|_2 + 3\sqrt{s} \| \widehat{\mathbf{f}} - R_s^{opt}(\widehat{f}) \|_1 \tag{3.27}$$

where $R_s^{opt}(\widehat{f})$ stands for the optimal $s-$term representation of $\widehat{\mathbf{f}}$. For this deterministic SFT algorithm, the runtime and the number of measurements used are both $\mathcal{O}(s^2 \log^4 N)$.

It is worth to mention that one can easily change the above deterministic SFT algorithm into a randomized algorithm by decreasing the number of co-prime numbers used in the algorithm. Actually, it has been showed in [2] and [4] that by choosing $\mathcal{O}(\log(\frac{N}{1-\sigma}))$ co-prime numbers in the SFT algorithm in [33], we will find the correct recovery with probability at least σ. Both the sampling complexity and runtime will be $\mathcal{O}(s \log^3(N) \cdot \log\left(\frac{N}{1-\sigma}\right))$. We encourage the reader to go to [33] [2] [4] and [8] for more details.

Chapter 4

Approach Based On filter function and fast discrete convolution

From now on, let us assume the input signal $\mathbf{f}$ is a length N vector which contains N equally spaced samples $f(\frac{2\pi j}{N})$ of the signal function, where $j = (-\frac{N}{2}, \frac{N}{2}] \cap \mathbb{Z}$. Assume that there exist a SFT algorithm which requires m samples $\{f(x_k)\}_{k=1}^{m}$ to find the best s-term approximation of the signal. However, the given input vector $\mathbf{f}$ only contains N equally spaced samples of f, and so does not necessarily contains all the samples the SFT algorithm needs. As a consequence, we need to try to interpolate these required function values $\{f(x_k)\}_{k=1}^{m}$ from $\mathbf{f}$.

There are two requirements for an ideal interpolation method: fast and accurate. Keep in mind that we are trying to create a sub-linear algorithm which can outperform some existing Fourier transform algorithms (e.g. FFTW) in the sense of speed and at the mean time provide good approximation to the sparse signal. The most straight forward idea is to use polynomial interpolation. However, we prove in section 4.1 that polynomial interpolation is poorly conditioned with high oscillate functions. One will need large number of points to reduce the error. In another word, polynomial interpolation is fast and easy to implement but not accurate enough. Then we use the rest of this chapter to introduce our approach based in filter function and fast discrete convolution. In section 4.2, we introduce the filter function that used in our approach: periodic Gaussian function. We then point out that periodic Gaussian function is not the only possible choice, however, it is a very good candidate because it has relatively simple analytic form and can be numerical evaluated easily. Then in section 4.3 we give the description of our proposed approach in the form of pseudo code. At the very end, in section 4.4, we show that the value of function at desired point can be evaluated rapidly and accurately by using the filter function and fast discrete convolution approach.

4.1 Polynomial interpolation

In this section, we first consider an simple and intuitive method, e.g. polynomial interpolation, to get the desired function value at points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$. Polynomial interpolation is easy to implement and can be evaluated rapidly. However, by using the results from [5], we get the modified Lemma 2 which shows that the value of the desired points $\{x_k\}_{k=1}^m$ can only be evaluated accurately with polynomial interpolation when all the high frequencies of the signal function vanished. More precisely, polynomial interpolation can only works if all the frequencies ω with $|\omega| \geq \frac{N}{2\pi}$ have zero coefficients. This is equivalent to oversampling the signal function with a factor of π. We present the details of the result in the lemma below.

Lemma 2. *Let $x \in (-\pi, \pi]$ and $y_j = -\pi + \frac{2\pi j}{N}$ for $j = 0, ..., N-1$. Set $j' := \arg\min_j \left|x - y_j\right|$. Suppose that $\widehat{\mathbf{f}}_\omega = 0$ for all $|\omega| \geq \frac{N}{2\pi}$. Let $P(x)$ denote the value of polynomial interpolation by using the value of 2κ terms in $\mathbf{f}$ nearest to x, i.e. $\{f(x_{j'+j})\}_{j=-\kappa+1}^{\kappa}$. Then we have*

$$|f(x) - P(x)| < \epsilon \tag{4.1}$$

when $\kappa = \lceil \log_2(\frac{2\|\widehat{f}\|_1}{\epsilon}) \rceil$.

Proof. Let us take 2κ terms nearest to the desired point x and calculate the real and imaginary part of polynomial interpolation $P(x)$ separately. Let P_R denote the real part of the polynomial interpolation, and P_I be the imaginary part.

Note that

$$\begin{aligned}
|Re\{f(x)\} - P_R(x)| &= \frac{f^{(2\kappa)}(\epsilon)}{(2\kappa)!} \prod_{m=-\kappa+1}^{\kappa} (x_{j'} - \frac{2\pi(j'+m)}{N}) \\
&\leq \frac{\| f^{(2\kappa)} \|_\infty}{(2\kappa)!} \prod_{m=1}^{\kappa} (\frac{m2\pi}{N})^2 \\
&= \frac{\| f^{(2\kappa)} \|_\infty}{(2\kappa)!} (\frac{2\pi}{N})^{2\kappa} \prod_{m=1}^{\kappa} m^2 \\
&= \| f^{(2\kappa)} \|_\infty (\frac{2\pi}{N})^{2\kappa} \frac{(\kappa!)^2}{(2\kappa)!} \\
&\leq \frac{\| f^{(2\kappa)} \|_\infty}{2^\kappa} (\frac{2\pi}{N})^{2\kappa}
\end{aligned}$$

The first equation is Lagrange interpolation error. Also we notice that for $\| f^{(2\kappa)} \|_\infty$ we have:

$$\begin{aligned}
\| f^{(2\kappa)} \|_\infty &= | \sum_{|\omega|<\frac{N}{2\pi}} \widehat{f}(\omega)(i\omega)^{2\kappa} e^{i\omega\bar{x}}| \\
&\leq \sum_{|\omega|<\frac{N}{\pi}} |\widehat{f}(\omega)| (\frac{N}{2\pi})^{2\kappa} \\
&\leq \| \widehat{f} \|_1 (\frac{N}{2\pi})^{2\kappa}
\end{aligned}$$

Plug this into previous inequality we get:

$$|Re\{f(x)\} - P_R(x)| \leq \frac{\| \widehat{f} \|_1}{2^\kappa}$$

Set $\kappa = \lceil log_2(\frac{2\|\widehat{f}\|_1}{\epsilon})\rceil$, we have $|Re\{f(x)\} - P_R(x)| \leq \frac{\epsilon}{2}$.

Same argument imply imagine part also has $|Im\{f(x)\} - P_I(x)| \leq \frac{\epsilon}{2}$. □

From above Lemma, we can see that in order to get accurate estimation from the polynomial interpolation, we need to oversampling the signal function f with a factor of π, which is an unpleasant assumption. The reason polynomial interpolation method not working very well is because the high order derivative grows exponentially, which makes the polynomial interpolation to be poorly conditioned with high oscillate functions. A very natural idea is to try to control or eliminate the high frequency terms in Fourier space. Filter function is an ideal choice in this task.

In the next few sections, we try to solve this problem by showing that with appropriate filter function (e.g. periodic Gaussian function), one only needs very few samples $\mathcal{O}(\log N)$ to estimate the desired points accurately and rapidly.

4.2 Periodic Gaussian Function

The filter function plays a crucial role in our approach. Later (in section 4.4) we will show that a good filter function can help us to evaluate the value of the function at desired points rapidly and accurately. Before we start to discuss what properties we expect to see in the filter function, let us first give the definition of effective support.

Definition 1. *Assume X is the domain of a function f. Then for any positive number ϵ,* *the effective support of f according to ϵ is:*

$$X_\epsilon(f) = \{x \in X \mid |f(x)| > \epsilon\}$$

Intuitively, we want our filter function to have wide effective support in Fourier space, so that we only need to shift it a few times in order to cover the whole bandwidth. And at the same time, we want it to have narrow effective support in time space. The benefit of these two properties will be discussed in more detail later. The good news is that the uncertainty theorem of Heisenberg tells us that the balance has to be reached between the time and frequency resolution. More precisely, if we let σ_t and σ_ω denote the root deviation of a function f in time space and Fourier space respectively. Then we have

$$\sigma_t \sigma_\omega \geq \frac{1}{4} \tag{4.2}$$

This means if a function has large variance in Fourier space, then it must have small variance in time space, which is exactly what we want.

There are a lot of different kinds of filter functions available in the market. For example, rectangular window function (sometimes known as the boxcar or Dirichlet window), Prolate spherical functions, Kiaser Bessel functions and C_∞ bump function, etc. Theoretically speaking, they all can be used in our method. However, we choose to use periodic Gaussian function in our approach because of two reasons: first, it has simple analytic form which makes it easier to control its variance in time and Fourier space; and second, which is also the most important part, periodic Gaussian function is easy for a computer to evaluate. Next, we give the definition of periodic Gaussian function in Lemma 4.3 and then show that we can always get the desired effective support in both time and Fourier space by tuning its parameters.

In the sections that follow the 2π−periodic Gaussian $g : [-\pi, \pi] \to \mathbb{R}^+$ defined by

$$g(x) = \frac{1}{c_1} \sum_{n=-\infty}^{\infty} \mathrm{e}^{-\frac{(x-2n\pi)^2}{2c_1^2}} \tag{4.3}$$

with $c_1 \in \mathbb{R}^+$ will play a special role. The following lemmas recall several useful facts concerning both its decay, and its Fourier series coefficients.

Lemma 3. *The 2π−periodic Gaussian $g : [-\pi, \pi] \to \mathbb{R}^+$ has*

$$g(x) \leq \left(\frac{3}{c_1} + \frac{1}{\sqrt{2\pi}}\right) \mathrm{e}^{-\frac{x^2}{2c_1^2}}$$

for all $x \in [-\pi, \pi]$.

Lemma 4. *The 2π−periodic Gaussian $g : [-\pi, \pi] \to \mathbb{R}^+$ has*

$$\widehat{g}_\omega = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{c_1^2 \omega^2}{2}}$$

for all $\omega \in \mathbb{Z}$. Thus, $\widehat{g} = \{\widehat{g}_\omega\}_{\omega \in \mathbb{Z}} \in \ell^2$ decreases monotonically as $|\omega|$ increases, and also has $\|\widehat{g}\|_\infty = \frac{1}{\sqrt{2\pi}}$.

Lemma 5. *Choose any $\tau \in \left(0, \frac{1}{\sqrt{2\pi}}\right)$, $\alpha \in \left[1, \frac{N}{\sqrt{\ln N}}\right]$, and $\beta \in \left(0, \alpha\sqrt{\frac{\ln(1/\tau\sqrt{2\pi})}{2}}\right]$. Let $c_1 = \frac{\beta\sqrt{\ln N}}{N}$ in the definition of the periodic Gaussian g from (4.3). Then $\widehat{g}_\omega \in \left[\tau, \frac{1}{\sqrt{2\pi}}\right]$ for all $\omega \in \mathbb{Z}$ with $|\omega| \leq \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil$.*

The proofs of lemmas 3, 4, and 5 are included in Appendix A for the sake of completeness. Intuitively, we will utilize the periodic function g from (4.3) as a bandpass filter below.

Looking at lemma 5 in this context we can see that its parameter τ will control the effect of $\widehat{g}$ on the frequency passband defined by its parameter α. Deciding on the two parameters τ, α then constrains β which, in turn, fixes the periodic Gaussian g by determining its constant coefficient c_1. As we shall see, the parameter β will also determine the speed and accuracy with which we can approximately sample (i.e., evaluate) the function $f * g$. For this reason it will become important to properly balance these parameters against one another in subsequent sections.

4.3 Description of the Proposed Approach

In this section we give the reader a general idea about the structure of our discrete SFT approach. Let us assume that we have access to an SFT algorithm $\mathcal{A}$ which requires m function evaluations of a 2π-periodic function $f : [-\pi, \pi] \to \mathbb{C}$ in order to produce an s-sparse approximation to $\widehat{f}$. For any non-adaptive SFT algorithm $\mathcal{A}$ the m points $\{x_k\}_{k=1}^{m} \subset [-\pi, \pi]$ at which $\mathcal{A}$ needs to evaluate f can be determined before $\mathcal{A}$ is actually executed. As a result, the function evaluations $\{f(x_k)\}_{k=1}^{m}$ required by $\mathcal{A}$ can also be evaluated before $\mathcal{A}$ is ever run. Indeed, if the SFT algorithm $\mathcal{A}$ is both nonadaptive *and* robust to noise it suffices to *approximate* the function evaluations $\{f(x_k)\}_{k=1}^{m}$ required by $\mathcal{A}$ before it is executed.[1] These simple ideas form the basis for the proposed computational approach outlined in Algorithm 1.

The objective of Algorithm 1 is to use a nonadaptive and noise robust SFT algorithm $\mathcal{A}$ which requires off-grid function evaluations in order to approximately compute the DFT of a given vector $\mathbf{f} \in \mathbb{C}^N$, $\hat{\mathbf{f}} = F\mathbf{f}$. Note that computing $\hat{\mathbf{f}}$ is equivalent to computing

[1] We hasten to point out, moreover, that similar ideas can also be employed for *adaptive* and noise robust SFT algorithms in order to approximately evaluate f in an "on demand" fashion as well. We leave the details to the interested reader.

Algorithm 1: A Generic Method for Discretizing a Given SFT Algorithm $\mathcal{A}$

```
  Input  : Pointer to vector $\mathbf{f} \in \mathbb{C}^N$, sparsity $s \leq N$, nodes $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ at
           which the given SFT algorithm $\mathcal{A}$ needs function evaluations, and $\alpha, \beta$
           satisfying lemma 5
  Output: $\widehat{R}^s$, a sparse approximation of $\hat{\mathbf{f}} \in \mathbb{C}^N$
1 Initialize $\widehat{R}, \widehat{R}^s \leftarrow \emptyset$
2 Set $c_1 = \frac{\beta\sqrt{\ln N}}{N}$ in the definition of periodic Gaussian $g$ from (4.3), and $c_2 = \frac{\alpha\sqrt{\ln N}}{2}$
3 for $j$ from 1 to $\lceil c_2 \rceil$ do
4     $q = -\left\lceil \frac{N}{2} \right\rceil + 1 + (2j-1)\left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil$
5     Modulate $g$ to be $\tilde{g}_q(x) := \mathbb{e}^{-\mathbb{i}qx} g(x)$
6     for each point $x \in \{x_k\}_{k=1}^m$ do
7         Use $\mathbf{f}$ to approximately compute $(\tilde{g}_q * f)(x)$ as per §4.4
8     end
9     Run given SFT algorithm $\mathcal{A}$ using the approximate function evaluations
      $\{(\tilde{g}_q * f)(x_k)\}_{k=1}^m$ in order to find an $s$-sparse Fourier approximation,
      $\widehat{R}_{temp} \subset \mathbb{Z} \times \mathbb{C}$, of $\widehat{\tilde{g}_q * f}$.
10    for each (frequency,Fourier coefficient) pair $(\omega, c_\omega) \in \widehat{R}_{temp}$ do
11        if $\omega \in \left[q - \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil, q + \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil\right) \cap B$ then
12            $\widehat{R} = \widehat{R} \cup \left\{\left(\omega, c_\omega / \left(\widehat{\tilde{g}_q}\right)_\omega\right)\right\}$
13        end
14    end
15 end
16 Choose the $s$ frequencies $\omega$ with $(\omega, \tilde{c}_\omega) \in \widehat{R}$ having the largest $|\tilde{c}_\omega|$, and put those
   $(\omega, \tilde{c}_\omega)$ in $\widehat{R}^s$
17 Return $\widehat{R}^s$
```

the Fourier series coefficients of the degree N trigonometric interpolant of $\mathbf{f}$. Hereafter the 2π-periodic function $f : [-\pi, \pi] \to \mathbb{C}$ under consideration will always be this degree N trigonometric interpolant of $\mathbf{f}$. Our objective then becomes to approximately compute $\widehat{f}$ using $\mathcal{A}$. Unfortunately, our given input vector $\mathbf{f}$ only contains equally spaced function evaluations of f, and so does not actually contain the function evaluations $\{f(x_k)\}_{k=1}^m$ required by $\mathcal{A}$. As a consequence, we are forced to try to interpolate these required function evaluations $\{f(x_k)\}_{k=1}^m$ from the available equally spaced function evaluations $\mathbf{f}$.

Directly interpolating the required function evaluations $\{f(x_k)\}_{k=1}^m$ from $\mathbf{f}$ for an arbitrary degree N trigonometric polynomial f using standard techniques appears to be either too inaccurate, or else too slow to work well in our setting.[2] As a result, Algorithm 1 instead uses $\mathbf{f}$ to rapidly approximate samples from the convolution of the unknown trigonometric polynomial f with (several modulations of) a known filter function g. Thankfully, all of the evaluations $\{(g * f)(x_k)\}_{k=1}^m$ can be approximated very accurately using only the data in $\mathbf{f}$ in just $\mathcal{O}(m \log N)$-time when g is chosen carefully enough (see §4.4 below). The given SFT algorithm $\mathcal{A}$ is then used to approximate the Fourier coefficients of $g * f$ for each modulation of g using these approximate evaluations. Finally, $\hat{\mathbf{f}}$ is then approximated using the recovered sparse approximation for each $\widehat{g * f}$ combined with our a priori knowledge of $\widehat{g}$.

[2]Each function evaluation $f(x_k)$ needs to be accurately computed in just $\mathcal{O}(\log^c N)$-time in order to allow us to achieve our overall desired runtime for Algorithm 1.

4.4 Rapidly and Accurately Evaluating the convolution

In this section we will carefully consider the approximation of $(f * g)(x)$ by a *severely truncated version* of the semi-discrete convolution sum

$$\frac{1}{N}\sum_{j=0}^{N-1} f\left(-\pi+\frac{2\pi j}{N}\right) g\left(x+\pi-\frac{2\pi j}{N}\right) \tag{4.4}$$

for any given value of $x \in [-\pi,\pi]$. Our goal is to determine exactly how many terms of this finite sum we actually need in order to obtain an accurate approximation of $f * g$ at an arbitrary x-value. More specifically, we aim to use as few terms from this sum as absolutely possible in order to ensure, e.g., an approximation error of size $\mathcal{O}(N^{-2})$.

Without loss of generality, let us assume that $N = 2M+1$ is odd – this allows us to express B, the set of N Fourier modes about zero, as

$$B := \left(-\left\lceil\frac{N}{2}\right\rceil, \left\lfloor\frac{N}{2}\right\rfloor\right] \cap \mathbb{Z} = [-M, M] \cap \mathbb{Z}.$$

In the lemmas and theorems below the function $f : [-\pi,\pi] \to \mathbb{C}$ will always denote a degree-N trigonometric polynomial of the form

$$f(x) = \sum_{\omega\in B} \widehat{f}_{\omega} \mathbb{e}^{\mathbb{i}\omega x}.$$

Furthermore, g will always denote the periodic Gaussian as defined above in (4.3). Finally, we will also make use of the Dirichlet kernel $D_M : \mathbb{R} \to \mathbb{C}$, defined by

$$D_M(y) = \frac{1}{2\pi}\sum_{n=-M}^{M} \mathbb{e}^{\mathbb{i}ny} = \frac{1}{2\pi}\sum_{n\in B} \mathbb{e}^{\mathbb{i}ny}.$$

The relationship between trigonometric polynomials such as f and the Dirichlet kernel D_M is the subject of the following lemma.

Lemma 6. *Let $h : [-\pi, \pi] \to \mathbb{C}$ have $\widehat{h}_\omega = 0$ for all $\omega \notin B$, and define the set of points $\{y_j\}_{j=0}^{2M} = \left\{-\pi + \frac{2\pi j}{N}\right\}_{j=0}^{2M}$. Then,*

$$2\pi\left(h * D_M\right)(x) \;=\; h(x) \;=\; \frac{2\pi}{N}\sum_{j=0}^{2M} h\left(y_j\right) D_M\left(x - y_j\right)$$

holds for all $x \in [-\pi, \pi]$.

Proof. By the definition of D_M, we trivially have $2\pi\left(\widehat{D_M}\right)_\omega = \chi_B(\omega)\ \forall \omega \in \mathbb{Z}$. Thus,

$$\widehat{h} = 2\pi \cdot \widehat{h} \circ \widehat{D_M} = 2\pi \cdot \widehat{h * D_M}$$

where, as before, $\circ$ denotes the Hadamard product, and $*$ denotes convolution. This yields $h(x) = 2\pi\left(h * D_M\right)(x)$ and so establishes the first equality above. To establish the second equality above, recall from (2.1) that for any $\omega \in B$ we will have

$$\widehat{h}_\omega = \frac{(-1)^\omega}{N}\sum_{j=0}^{2M} h\left(-\pi + \frac{2\pi j}{N}\right)\mathbb{e}^{\frac{-2\pi \mathbb{i} j \omega}{N}} = \frac{1}{N}\sum_{j=0}^{2M} h\left(y_j\right)\mathbb{e}^{-\mathbb{i}\omega y_j},$$

since h is a trigonometric polynomial. Thus, given $x \in [-\pi, \pi]$ one has

$$h(x) = \sum_{\omega \in B}\widehat{h}_\omega \mathbb{e}^{\mathbb{i}\omega x} = \frac{1}{N}\sum_{j=0}^{2M}\left(h\left(y_j\right)\sum_{\omega \in B}\mathbb{e}^{\mathbb{i}\omega\left(x - y_j\right)}\right) = \frac{2\pi}{N}\sum_{j=0}^{2M} h\left(y_j\right) D_M\left(x - y_j\right).$$

We now have the desired result. $\square$

We can now write a formula for $g * f$ which only depends on N evaluations of f in $[-\pi, \pi]$.

Lemma 7. *Given the set of equally spaced points* $\{y_j\}_{j=0}^{2M} = \left\{-\pi + \frac{2\pi j}{N}\right\}_{j=0}^{2M}$ *one has that*

$$(g * f)(x) = \frac{1}{N}\sum_{j=0}^{2M} f\left(y_j\right)\int_{-\pi}^{\pi} g\left(x - u - y_j\right) D_M(u)\, du$$

for all $x \in [-\pi, \pi]$.

Proof. By Lemma 6, we have

$$\begin{aligned}(g * f)(x) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} g(x-y) f(y)\, dy = \frac{1}{N}\int_{-\pi}^{\pi} g(x-y)\sum_{j=0}^{2M} f\left(y_j\right) D_M\left(y - y_j\right) dy \\ &= \frac{1}{N}\sum_{j=0}^{2M} f\left(y_j\right)\int_{-\pi}^{\pi} g\left(x - u - y_j\right) D_M(u)\, du.\end{aligned}$$

The last equality holds after a change of variables since g and D_M are both $2\pi-$periodic. □

The next two lemmas will help us bound the error produced by discretizing the integral weights present in the finite sum provided by Lemma 7 above. More specifically, they will ultimately allow us to approximate the sum in Lemma 7 by the sum in (4.4).

Lemma 8. *Let* $x \in [-\pi, \pi]$ *and* $y_j = -\pi + \frac{2\pi j}{N}$ *for some* $j = 0, \ldots, 2M$*. Then,*

$$\int_{-\pi}^{\pi} g\left(x - u - y_j\right) D_M(u)\, du = \sum_{n \in B} \widehat{g}_n \mathbb{e}^{\mathbb{i} n\left(x - y_j\right)}.$$

Proof. Recalling that $2\pi\left(\widehat{D_M}\right)_\omega = \chi_B(\omega)$ for all $\omega \in \mathbb{Z}$ we have that

$$\begin{aligned}\int_{-\pi}^{\pi} g\left(x - u - y_j\right) D_M(u)\, du = 2\pi\left(D_M * g\right)\left(x - y_j\right) &= \sum_{n \in \mathbb{Z}} \widehat{g}_n \chi_B(n)\, \mathbb{e}^{\mathbb{i} n\left(x - y_j\right)} \\ &= \sum_{n \in B} \widehat{g}_n \mathbb{e}^{\mathbb{i} n\left(x - y_j\right)}.\end{aligned}$$

□

Lemma 9. *Denote* $I(a) := \int_{-a}^{a} e^{-x^2} dx$ *for* $a > 0$*; then*

$$\pi\left(1 - e^{-a^2}\right) < I^2(a) < \pi\left(1 - e^{-2a^2}\right).$$

Proof. Let $a > 0$ and observe that

$$I^2(a) = \int_{-a}^{a}\int_{-a}^{a} e^{-x^2-y^2}dxdy > \iint_{\left\{x^2+y^2\le a^2\right\}} e^{-\left(x^2+y^2\right)}dxdy = \pi\left(1 - e^{-a^2}\right).$$

The first equality holds by Fubini's theorem, and the inequality follows simply by integrating a positive function over a disk of radius a as opposed to a square of sidelength $2a$. A similar argument yields the upper bound. □

We are now ready to bound the difference between the integral weights present in the finite sum provided by Lemma 7, and the $g\left(x - y_j\right)$-weights present in the sum (4.4).

Lemma 10. *Choose any* $\tau \in \left(0, \frac{1}{\sqrt{2\pi}}\right)$, $\alpha \in \left[1, \frac{N}{\sqrt{\ln N}}\right]$, *and* $\beta \in \left(0, \alpha\sqrt{\frac{\ln\left(1/\tau\sqrt{2\pi}\right)}{2}}\,\right]$. *Let* $c_1 = \frac{\beta\sqrt{\ln N}}{N}$ *in the definition of the periodic Gaussian* g *so that*

$$g(x) = \frac{N}{\beta\sqrt{\ln N}}\sum_{n=-\infty}^{\infty} e^{-\frac{(x-2n\pi)^2N^2}{2\beta^2\ln N}}.$$

Then for all $x \in [-\pi, \pi]$ *and* $y_j = -\pi + \frac{2\pi j}{N}$,

$$\left|g\left(x - y_j\right) - \int_{-\pi}^{\pi} g\left(x - u - y_j\right) D_M(u)\, du\right| < \frac{N^{1-\frac{\beta^2}{18}}}{\beta\sqrt{\ln N}}.$$

Proof. Using Lemma 8 we calculate

$$\begin{aligned}\left|g\left(x-y_j\right)-\int_{-\pi}^{\pi} g\left(x-u-y_j\right) D_M(u)\, du\right| &= \left|g\left(x-y_j\right)-\sum_{n\in B}\widehat{g}_n e^{in\left(x-y_j\right)}\right| \\ &= \left|\sum_{n\in B^c}\widehat{g}_n e^{in\left(x-y_j\right)}\right| \\ &\le \frac{1}{\sqrt{2\pi}}\sum_{|n|>M} e^{-\frac{c_1^2 n^2}{2}} \qquad \text{(Using Lemma 4)} \\ &\le \frac{2}{\sqrt{2\pi}}\int_M^{\infty} e^{-\frac{c_1^2 n^2}{2}}\, dn \\ &= \sqrt{\frac{2}{\pi}}\int_M^{\infty} e^{-\frac{\beta^2 n^2 \ln N}{2N^2}}\, dn.\end{aligned}$$

Upon the change of variable $v = \frac{\beta n \sqrt{\ln N}}{\sqrt{2}N}$, we get that

$$\begin{aligned} &\left|g\left(x-y_j\right)-\int_{-\pi}^{\pi} g\left(x-u-y_j\right) D_M(u)\, du\right| \\ \le\ & \sqrt{\frac{2}{\pi}}\frac{\sqrt{2}N}{\beta\sqrt{\ln N}}\int_{\frac{\beta M\sqrt{\ln N}}{\sqrt{2}N}}^{\infty} e^{-v^2}\, dv \\ =\ & \frac{2N}{\beta\sqrt{\pi\ln N}}\frac{1}{2}\left(\int_{-\infty}^{\infty} e^{-v^2}\, dv - \int_{-\frac{\beta M\sqrt{\ln N}}{\sqrt{2}N}}^{\frac{\beta M\sqrt{\ln N}}{\sqrt{2}N}} e^{-v^2}\, dv\right) \\ <\ & \frac{N}{\beta\sqrt{\pi\ln N}}\left(\sqrt{\pi}-\sqrt{\pi\left(1-e^{-\frac{\beta^2 M^2 \ln N}{2N^2}}\right)}\right) \\ =\ & \frac{N}{\beta\sqrt{\ln N}}\left(1-\sqrt{1-N^{-\frac{\beta^2 M^2}{2N^2}}}\right)\end{aligned}$$

where the last inequality follows from Lemma 9. Noting now that

$$y\in[0,1] \implies 1-\sqrt{1-y}\le y,$$

and that $\frac{N}{M} = 2 + \frac{1}{M} \in (2, 3]$ for all $M \in \mathbb{Z}^+$, we can further see that

$$\frac{N}{\beta\sqrt{\ln N}}\left(1 - \sqrt{1 - N^{-\frac{\beta^2 M^2}{2N^2}}}\right) \leq \frac{N}{\beta\sqrt{\ln N}} N^{-\frac{\beta^2 M^2}{2N^2}} \leq \frac{N^{1-\frac{\beta^2}{18}}}{\beta\sqrt{\ln N}}$$

also always holds. □

With the lemmas above we can now prove that (4.4) can be used to approximate $(g * f)(x)$ for all $x \in [-\pi, \pi]$ with controllable error.

Theorem 2. *Let $p \geq 1$. Using the same values of the parameters from lemma 10 above, one has*

$$\left|(g * f)(x) - \frac{1}{N}\sum_{j=0}^{2M} f(y_j)\, g(x - y_j)\right| \leq \frac{\|\mathbf{f}\|_p}{\beta\sqrt{\ln N}} N^{1-\frac{\beta^2}{18}-\frac{1}{p}}$$

for all $x \in [-\pi, \pi]$.

Proof. Using lemmas 7 and 10 followed by Holder's inequality, we have

$$\begin{aligned} & \left|(g * f)(x) - \frac{1}{N}\sum_{j=0}^{2M} f(y_j)\, g(x - y_j)\right| \\ = & \left|\frac{1}{N}\sum_{j=0}^{2M} f(y_j)\left(g(x - y_j) - \int_{-\pi}^{\pi} g(x - u - y_j)\, D_M(u)\, du\right)\right| \\ \leq & \frac{1}{N}\sum_{j=0}^{2M} \left|f(y_j)\right| \frac{N^{1-\frac{\beta^2}{18}}}{\beta\sqrt{\ln N}} \leq \frac{N^{\frac{-\beta^2}{18}}}{\beta\sqrt{\ln N}} \|\mathbf{f}\|_p N^{1-\frac{1}{p}}. \end{aligned}$$

□

To summarize, Theorem 2 tells us that $(g * f)(x)$ can be approximately computed in $\mathcal{O}(N)$-time for any $x \in [-\pi, \pi]$ using (4.4). This linear runtime cost may be reduced significantly, however, if one is willing to accept an additional trade-off between accuracy and the number of terms needed in the sum (4.4). This trade-off is characterized in the next lemma.

Lemma 11. *Let $x \in [-\pi, \pi]$, $p \geq 1$, $\gamma \in \mathbb{R}^+$, and $\kappa := \lceil \gamma \ln N \rceil + 1$. Set $j' := \arg\min_j |x - y_j|$. Using the same values of the other parameters from lemma 10 above, one has*

$$\left| \frac{1}{N} \sum_{j=0}^{2M} f(y_j)\, g(x - y_j) - \frac{1}{N} \sum_{j=j'-\kappa}^{j'+\kappa} f(y_j)\, g(x - y_j) \right| \leq 2 \|\mathbf{f}\|_p \; N^{-\frac{2\pi^2 \gamma^2}{\beta^2}}$$

for all $\beta \geq 4$ and $N \geq \beta^2$.

Proof. Appealing to lemma 3 and recalling that $c_1 = \frac{\beta \sqrt{\ln N}}{N}$ we can see that

$$g(x) \leq \left(\frac{3N}{\beta \sqrt{\ln N}} + \frac{1}{\sqrt{2\pi}} \right) \mathrm{e}^{-\frac{x^2 N^2}{2\beta^2 \ln N}}.$$

Using this fact we have that

$$g\left(x - y_{j' \pm k}\right) \leq \left(\frac{3N}{\beta \sqrt{\ln N}} + \frac{1}{\sqrt{2\pi}} \right) \mathrm{e}^{-\frac{\left(x - y_{j'\pm k}\right)^2 N^2}{2\beta^2 \ln N}} \leq \left(\frac{3N}{\beta \sqrt{\ln N}} + \frac{1}{\sqrt{2\pi}} \right) \mathrm{e}^{-\frac{(2k-1)^2 \pi^2}{2\beta^2 \ln N}}$$

for all $k \in \mathbb{Z}_N$. As a result, one can now bound

$$\left| \frac{1}{N} \sum_{j=0}^{2M} f(y_j)\, g(x - y_j) - \frac{1}{N} \sum_{j=j'-\kappa}^{j'+\kappa} f(y_j)\, g(x - y_j) \right|$$

above by

$$\left(\frac{3}{\beta \sqrt{\ln N}} + \frac{1}{N\sqrt{2\pi}} \right) \sum_{k=\kappa+1}^{N-2\kappa-1} \left(\left| f\left(y_{j'-k}\right) \right| + \left| f\left(y_{j'+k}\right) \right| \right) \mathrm{e}^{-\frac{(2k-1)^2 \pi^2}{2\beta^2 \ln N}}, \qquad (4.5)$$

where the y_j-indexes are considered modulo N as appropriate.

Our goal is now to employ Holder's inequality on (4.5). Toward that end, we will now bound the q-norm of the vector $\mathbf{h} := \left\{ \mathbb{e}^{-\frac{\left(\kappa+\ell-\frac{1}{2}\right)^2 2\pi^2}{\beta^2 \ln N}} \right\}_{\ell=1}^{N-2\kappa-1}$. Letting $a := q\left(\frac{4}{\beta^2 \ln N}\right)$ we have that

$$\|\mathbf{h}\|_q^q = \sum_{\ell=1}^{N-2\kappa-1} \mathbb{e}^{-\frac{\pi^2}{2}\left(\kappa+\ell-\frac{1}{2}\right)^2 a} < \sum_{\ell=\kappa}^{\infty} \mathbb{e}^{-\frac{\pi^2}{2}\ell^2 a} \leq \int_{\kappa-1}^{\infty} \mathbb{e}^{-\frac{\pi^2 x^2}{2}a}\, dx$$
$$\leq \sqrt{\frac{1}{2\pi a}} - \frac{1}{\pi\sqrt{2a}} \int_{-\pi(\kappa-1)\sqrt{\frac{a}{2}}}^{\pi(\kappa-1)\sqrt{\frac{a}{2}}} \mathbb{e}^{-u^2}\, du \leq \sqrt{\frac{1}{2\pi a}} \mathbb{e}^{-\frac{a\pi^2}{2}(\kappa-1)^2} \leq \frac{\beta}{2}\sqrt{\frac{\ln N}{2\pi q}} N^{-\frac{2q\pi^2\gamma^2}{\beta^2}},$$

where we have used Lemma 9 once again. As a result we have that

$$\|\mathbf{h}\|_q \leq \left(\frac{\beta^2 \ln N}{8\pi}\right)^{\frac{1}{2q}} q^{-\frac{1}{2q}} N^{-\frac{2\pi^2\gamma^2}{\beta^2}} \leq \left(\frac{\beta^2 \ln N}{8\pi}\right)^{\frac{1}{2q}} N^{-\frac{2\pi^2\gamma^2}{\beta^2}}$$

for all $q \geq 1$. Applying Holder's inequality on (4.5) we can now see that (4.5) is bounded above by

$$2\left(\frac{3}{\beta\sqrt{\ln N}} + \frac{1}{N\sqrt{2\pi}}\right) \|\mathbf{f}\|_p \left(\frac{\beta^2 \ln N}{8\pi}\right)^{\frac{1}{2}-\frac{1}{2p}} N^{-\frac{2\pi^2\gamma^2}{\beta^2}}.$$

The result now follows. $\square$

We may now finally combine the truncation and estimation errors in Theorem 2 and Lemma 11 above in order to bound the total error one incurs by approximating $(g * f)(x)$ via a truncated portion of (4.4) for any given $x \in [-\pi, \pi]$.

Theorem 3. *Fix $x \in [-\pi, \pi]$, $p \geq 1$ (or $p = \infty$), $\frac{N}{36} \geq r \geq 1$, and $g : [-\pi, \pi] \to \mathbb{R}^+$ to be the $2\pi-$periodic Gaussian* (4.3) *with $c_1 := \frac{6\sqrt{\ln(N^r)}}{N}$. Set $j' := \arg\min_j |x - y_j|$ where*

$y_j = -\pi + \frac{2\pi j}{N}$ for all $j = 0, \dots, 2M$. Then,

$$\left| (g * f)(x) - \frac{1}{N} \sum_{j = j' - \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil - 1}^{j' + \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil + 1} f\left(y_j\right) g\left(x - y_j\right) \right| \leq 3 \frac{\|\mathbf{f}\|_p}{N^r}.$$

As a consequence, we can see that $(g * f)(x)$ can always to computed to within $\mathcal{O}\left(\|\mathbf{f}\|_\infty N^{-r}\right)$-error in just $\mathcal{O}(r \log N)$-time for any given $\mathbf{f} \in \mathbb{C}^N$ once the $\left\{g\left(x - y_j\right)\right\}_{j = j' - \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil - 1}^{j' + \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil + 1}$ have been precomputed.

Proof. Combining Theorem 2 and Lemma 11 we can see that

$$\begin{aligned} &\left| (g * f)(x) - \frac{1}{N} \sum_{j = j' - \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil - 1}^{j' + \left\lceil \frac{6r}{\sqrt{2}\pi} \ln N \right\rceil + 1} f\left(y_j\right) g\left(x - y_j\right) \right| \\ \leq \quad & \|\mathbf{f}\|_p \left(\frac{1}{\beta \sqrt{\ln N}} N^{1 - \frac{\beta^2}{18} - \frac{1}{p}} + 2\ N^{-\frac{2\pi^2 \gamma^2}{\beta^2}} \right) \end{aligned}$$

where $\beta = 6\sqrt{r} \geq 6$, $N \geq 36r = \beta^2$, and $\gamma = \frac{6r}{\sqrt{2}\pi} = \frac{\beta\sqrt{r}}{\sqrt{2}\pi}$. $\square$

We are now prepared to bound the error of the proposed approach when utilizing the SFTs developed in [4].

Chapter 5

One Fully Discrete SFT algorithm

In this Chapter, we use the method described in Chapter 4 to create a fully discrete SFT algorithm based on the Algorithm 3 in [4]. More precisely, we replace the $\mathcal{A}$ in line 9 of Algorithm 1 with the SFT Algorithm 3 in [4]. As mentioned in the introduction, the Algorithm 3 in [4] is a non-adaptive SFT algorithm, which means the m points $\{x_k\}_{k=1}^{m} \subset [-\pi, \pi]$ at which Algorithm 3 needs to evaluate f can be determined before the Algorithm 3 is actually executed. One thing that should be mentioned here is that $\mathcal{A}$ can also be replaced by an adaptive SFT algorithm, in which case the value of desired samples cannot be pre-computed, but has to be approximated on demand. There is, however, one requirement of the SFT algorithm used to replace $\mathcal{A}$ in line 9, that is the SFT algorithm has to be noise robust. This is because even though from previous chapter we know that we can always approximate the value of convolution $\tilde{g}_q * f$ with desired accuracy, the value of the sample is still not equal to the true value. In that case, the approximated values can be seen as the contaminated true function value $\{f(x_k) + n_k\}_{k=1}^{m}$. Now it is clear that the SFT algorithm has to be robust to noise if we want it to find the support of a signal function in Fourier space correctly by using the contaminated samples.

The reminder of this chapter is organized as follows. In section 5.1 we prove that the Algorithm 3 in [4] is robust to noise. To do that, first we generalize the Lemma 6 in [4] by considering the noisy signal samples. Then we prove the noise robust variant of Theorem 7 from §5 of [4]. This is the necessary prerequisites to use the Algorithm 3 in [4] to replace the $\mathcal{A}$ in Algorithm 1. Then in section 5.2 we give an error guarantee for Algorithm 1 when using the SFTs proposed in [4]. We also provide the runtime complexity for both deterministic and randomized version of our new fully discrete SFT algorithm.

5.1 On the Robustness of the SFTs proposed in [4]

The sparse Fourier transforms presented in [4] include both deterministic and randomized methods for approximately computing the Fourier series coefficients of a given 2π-periodic function f from its evaluations at m-points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$. The following results describe how accurate these algorithms will be when they are only given approximate evaluations of f at these points instead. These results are necessary because we will want to execute the SFTs developed in [4] on convolutions of the form $f * g$ below, but will only be able to approximately compute their values at each of the required points $x_1, \dots, x_m \in [-\pi, \pi]$.

Lemma 12. *Let $s, \epsilon^{-1} \in \mathbb{N} \setminus \{1\}$ with $(s/\epsilon) \geq 2$, and $\mathbf{n} \in \mathbb{C}^m$ be an arbitrary noise vector. There exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ such that Algorithm 3 on page 72 of [4], when given access to the corrupted samples $\{f(x_k) + n_k\}_{k=1}^m$, will identify a subset $S \subseteq B = \left(\frac{N}{2}, \frac{N}{2}\right] \cap \mathbb{Z}$ which is guaranteed to contain all $\omega \in B$ with*

$$\left|\widehat{f}_\omega\right| > 4 \left(\frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty \right). \tag{5.1}$$

Furthermore, every $\omega \in S$ returned by Algorithm 3 will also have an associated Fourier series *coefficient estimate $z_\omega \in \mathbb{C}$ which is guaranteed to have*

$$\left| \widehat{f}_\omega - z_\omega \right| \leq \sqrt{2} \left(\frac{\epsilon \cdot \left\| \hat{\mathbf{f}} - \hat{\mathbf{f}}^{\rm opt}_{(s/\epsilon)} \right\|_1}{s} + \left\| \widehat{f} - \widehat{f}|_B \right\|_1 + \|\mathbf{n}\|_\infty \right). \tag{5.2}$$

Both the number of required samples, m, and Algorithm 3's operation count are

$$\mathcal{O}\left(\frac{s^2 \cdot \log^4(N)}{\log\left(\frac{s}{\epsilon}\right) \cdot \epsilon^2} \right). \tag{5.3}$$

If succeeding with probability $(1-\delta) \in [2/3, 1)$ is sufficient, and $(s/\epsilon) \geq 2$, the Monte Carlo variant of Algorithm 3 referred to by Corollary 4 on page 74 of [4] may be used. This Monte Carlo variant reads only a randomly chosen subset of the noisy samples utilized by the deterministic algorithm,

$$\{f(\tilde{x}_k) + \tilde{n}_k\}_{k=1}^{\tilde{m}} \subseteq \{f(x_k) + n_k\}_{k=1}^{m},$$

yet it still outputs a subset $S \subseteq B$ which is guaranteed to simultaneously satisfy both of the following properties with probability at least $1-\delta$:

(i) S will contain all $\omega \in B$ satisfying (5.1), and

(ii) all $\omega \in S$ will have an associated coefficient estimate $z_\omega \in \mathbb{C}$ satisfying (5.2).

Finally, both this Monte Carlo variant's number of required samples, $\tilde{m}$, as well as its operation count will also always be

$$\mathcal{O}\left(\frac{s}{\epsilon} \cdot \log^3(N) \cdot \log\left(\frac{N}{\delta}\right) \right). \tag{5.4}$$

Using the preceding lemma one can easily prove the following noise robust variant of Theorem 7 (and Corollary 4) from §5 of [4]. The proofs of both results are outlined in Appendix B for the sake of completeness.

Theorem 4. *Suppose $f : [-\pi, \pi] \to \mathbb{C}$ has $\widehat{f} \in \ell^1 \cap \ell^2$. Let $s, \epsilon^{-1} \in \mathbb{N} \setminus \{1\}$ with $(s/\epsilon) \geq 2$, and $\mathbf{n} \in \mathbb{C}^m$ be an arbitrary noise vector. Then, there exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ together with a simple deterministic algorithm $\mathcal{A} : \mathbb{C}^m \to \mathbb{C}^{4s}$ such that $\mathcal{A}\left(\{f(x_k) + n_k\}_{k=1}^m\right)$ is always guaranteed to output (the nonzero coefficients of) a degree $\leq N/2$ trigonometric polynomial $y_s : [-\pi, \pi] \to \mathbb{C}$ satisfying*

$$\|f - y_s\|_2 \leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\mathrm{opt}}\right\|_2 + \frac{22\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_{(s/\epsilon)}^{\mathrm{opt}}\right\|_1}{\sqrt{s}} + 22\sqrt{s}\left(\left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty\right). \tag{5.5}$$

Both the number of required samples, m, and the algorithm's operation count are always

$$\mathcal{O}\left(\frac{s^2 \cdot \log^4(N)}{\log\left(\frac{s}{\epsilon}\right) \cdot \epsilon^2}\right). \tag{5.6}$$

If succeeding with probability $(1-\delta) \in [2/3, 1)$ is sufficient, and $(s/\epsilon) \geq 2$, a Monte Carlo variant of the deterministic algorithm may be used. This Monte Carlo variant reads only a randomly chosen subset of the noisy samples utilized by the deterministic algorithm,

$$\{f(\tilde{x}_k) + \tilde{n}_k\}_{k=1}^{\tilde{m}} \subseteq \{f(x_k) + n_k\}_{k=1}^m,$$

yet it still outputs (the nonzero coefficients of) a degree $\leq N/2$ trigonometric polynomial, $y_s : [-\pi, \pi] \to \mathbb{C}$, that satisfies (B.9) *with probability at least $1 - \delta$. Both its number of* *required samples, $\tilde{m}$, as well as its operation count will always be*

$$\mathcal{O}\left(\frac{s}{\epsilon}\cdot\log^3(N)\cdot\log\left(\frac{N}{\delta}\right)\right). \tag{5.7}$$

We now have the necessary prerequisites in order to discuss our general strategy for constructing several new fully discrete SFTs.

5.2 An Error Guarantee for Algorithm 1 when Using the SFTs Proposed in [4]

Given the $2\pi-$periodic Gaussian $g : [-\pi,\pi] \to \mathbb{R}^+$ (4.3), consider the periodic modulation of g, $\tilde{g}_q : [-\pi,\pi] \to \mathbb{C}$, for any $q \in \mathbb{Z}$ defined by

$$\tilde{g}_q\left(x\right) = \mathbb{e}^{-\mathbb{i}qx} g\left(x\right).$$

One can see that

$$\tilde{g}_q\left(x\right) = \mathbb{e}^{-\mathbb{i}qx}\sum_{\omega=-\infty}^{\infty}\widehat{g}_\omega \mathbb{e}^{\mathbb{i}\omega x} = \sum_{\omega=-\infty}^{\infty}\widehat{g}_\omega \mathbb{e}^{\mathbb{i}(\omega-q)x} = \sum_{\tilde{\omega}=-\infty}^{\infty}\widehat{g}_{\tilde{\omega}+q}\mathbb{e}^{\mathbb{i}\tilde{\omega}x},$$

so that the Fourier series coefficients of $\tilde{g}_q$ are those of g, shifted by q; that is,

$$\left(\widehat{\tilde{g}_q}\right)_\omega = \widehat{g}_{\omega+q}.$$

In line 9 of Algorithm 1, we provide the SFT Algorithm in [4] with the approximate evaluations of $\left\{\left(\tilde{g}_q * f\right)(x_k)\right\}_{k=1}^m$, namely, $\left\{\left(\tilde{g}_q * f\right)(x_k) + n_k\right\}_{k=1}^m$, where, by Theorem 3, the perturbations n_k are bounded, for instance, by

$$|n_k| \leq 3\frac{\|f\|_\infty}{N^r} \ \forall \ k = 1, \ldots, m.$$

With this in mind, let us apply lemma 12 to the function $\tilde{g}_q * f$. We have the following lemma.

Lemma 13. *Let $s \in [2, N] \cap \mathbb{N}$, and $\mathbf{n} \in \mathbb{C}^m$ be the vector containing the total errors incurred by approximating $\tilde{g}_q * f$ via a truncated version of (4.4), as per Theorem 3. There exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ such that Algorithm 3 on page 72 of [4], when given access to the corrupted samples $\left\{\left(\tilde{g}_q * f\right)(x_k) + n_k\right\}_{k=1}^m$, will identify a subset $S \subseteq B$ which is guaranteed to contain all $\omega \in B$ with*

$$\left|\left(\widehat{\tilde{g}_q * f}\right)_\omega\right| > 4\left(\frac{1}{s} \cdot \left\|\widehat{\tilde{g}_q * f} - \left(\widehat{\tilde{g}_q * f}\right)_s^{\text{opt}}\right\|_1 + 3\|\mathbf{f}\|_\infty N^{-r}\right) =: 4\tilde{\delta}.$$

Furthermore, every $\omega \in S$ returned by Algorithm 3 will also have an associated Fourier series coefficient estimate $z_\omega \in \mathbb{C}$ which is guaranteed to have

$$\left|\left(\widehat{\tilde{g}_q * f}\right)_\omega - z_\omega\right| \leq \sqrt{2}\tilde{\delta}.$$

Next, we need to guarantee that the estimates of $\widehat{\tilde{g}_q * f}$ returned by Algorithm 3 of [4] will yield good estimates of $\widehat{f}$ itself. We have the following.

Lemma 14. *Let $s \in [2, N] \cap \mathbb{N}$. Given a $2\pi-$periodic function $f : [-\pi, \pi] \to \mathbb{C}$, the periodic*

Gaussian g, and any of its modulations $\tilde{g}_q(x) = \mathbb{e}^{-\mathbb{i}qx} g(x)$, one has

$$\left\| \widehat{\tilde{g}_q * f} - \left(\widehat{\tilde{g}_q * f} \right)_s^{\mathrm{opt}} \right\|_1 \leq \frac{1}{2} \left\| \widehat{f} - \widehat{f}_s^{\mathrm{opt}} \right\|_1 .$$

Proof. Recall the definition of $R_s^{\mathrm{opt}}\left(\widehat{f}\right)$ as the subset of B containing the s most energetic frequencies of $\widehat{f}$, and observe that

$$\frac{1}{2} \left\| \widehat{f} - \widehat{f}_s^{\mathrm{opt}} \right\|_1 = \frac{1}{2} \sum_{\omega \in B \setminus R_s^{\mathrm{opt}}\left(\widehat{f}\right)} \left| \widehat{f}_\omega \right| \geq \sum_{\omega \in B \setminus R_s^{\mathrm{opt}}\left(\widehat{f}\right)} \left| \left(\widehat{\tilde{g}_q} \right)_\omega \cdot \widehat{f}_\omega \right|$$

since, by lemma 4, $\widehat{g}_\omega < \frac{1}{2}$ for all ω, and consequently, $\left(\widehat{\tilde{g}_q} \right)_\omega = \widehat{g}_{\omega+q} < \frac{1}{2}$ for all ω. Moreover,

$$\sum_{\omega \in B \setminus R_s^{\mathrm{opt}}\left(\widehat{f}\right)} \left| \left(\widehat{\tilde{g}_q} \right)_\omega \cdot \widehat{f}_\omega \right| \geq \sum_{\omega \in B \setminus R_s^{\mathrm{opt}}\left(\widehat{\tilde{g}_q * f}\right)} \left| \left(\widehat{\tilde{g}_q} \right)_\omega \cdot \widehat{f}_\omega \right| = \left\| \widehat{\tilde{g}_q * f} - \left(\widehat{\tilde{g}_q * f} \right)_s^{\mathrm{opt}} \right\|_1 .$$

□

Let us combine the guarantees above into the following lemma.

Lemma 15. *Let $s \in [2, N] \cap \mathbb{N}$, and $\mathbf{n} \in \mathbb{C}^m$ be the vector containing the total errors incurred by approximating $\tilde{g}_q * f$ via a truncated version of (4.4), as per Theorem 3. There exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ such that Algorithm 3 on page 72 of [4], when given access to the corrupted samples $\left\{ \left(\tilde{g}_q * f \right)(x_k) + n_k \right\}_{k=1}^m$, will identify a subset $S \subseteq B$ which is guaranteed to contain all $\omega \in B$ with*

$$\left| \left(\widehat{\tilde{g}_q * f} \right)_\omega \right| > 4 \left(\frac{1}{2s} \cdot \left\| \widehat{f} - \widehat{f}_s^{\mathrm{opt}} \right\|_1 + 3 \left\| \mathbf{f} \right\|_\infty N^{-r} \right) =: 4\delta.$$

Furthermore, every $\omega \in S$ returned by Algorithm 3 will also have an associated Fourier series coefficient estimate $z_\omega \in \mathbb{C}$ which is guaranteed to have

$$\left| \left(\widehat{\tilde{g}_q} \right)_\omega \cdot \widehat{f}_\omega - z_\omega \right| \leq \sqrt{2}\delta.$$

The lemma above implies that for any choice of q in line 4 of Algorithm 1, we are guaranteed to find all $\omega \in \left[q - \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil, q + \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil \right) \cap B$ with

$$\left| \widehat{f}_\omega \right| > \max_{\tilde{\omega}} \frac{4\delta}{\left(\widehat{\tilde{g}_q} \right)_{\tilde{\omega}}} \geq \frac{4\delta}{\tau}$$

where α and τ are as defined in lemma 5. Moreover, the Fourier series coefficient estimates z_ω returned by Algorithm 3 will satisfy

$$\left| \widehat{f}_\omega - \frac{z_\omega}{\left(\widehat{\tilde{g}_q} \right)_\omega} \right| \leq \max_{\tilde{\omega}} \frac{\sqrt{2}\delta}{\left(\widehat{\tilde{g}_q} \right)_{\tilde{\omega}}} \leq \frac{\sqrt{2}\delta}{\tau}.$$

Following Theorem 3, which guarantees a decay of N^{-r} in the total approximation error, let us set $\beta = 6\sqrt{r}$ for $1 \leq r \leq \frac{N}{36}$. Recall from lemma 3 the choice of $\beta \in \left(0, \alpha\sqrt{\frac{\ln(1/\tau\sqrt{2\pi})}{2}}\right]$ where τ is to be chosen from $\left(0, \frac{1}{\sqrt{2\pi}}\right)$. Thus, we must choose $\alpha \in \left[1, \frac{N}{\sqrt{\ln N}}\right]$ so that

$$6\sqrt{r} \leq \alpha\sqrt{\frac{\ln\left(1/\tau\sqrt{2\pi}\right)}{2}} \iff \alpha \geq \frac{6\sqrt{2r}}{\ln\left(1/\tau\sqrt{2\pi}\right)}.$$

We may remove the dependence on τ simply by setting, e.g., $\tau = \frac{1}{3}$. Then $\alpha = \mathcal{O}\left(\sqrt{r}\right)$.

We are now ready to state the recovery guarantee of Algorithm 1 and its operation count.

Theorem 5. *Let $N \in \mathbb{N}$, $s \in [2, N] \cap \mathbb{N}$, and $1 \leq r \leq \frac{N}{36}$ as in Theorem 3. If Algorithm 3 of [4] is used in Algorithm 1 then Algorithm 1 will always deterministically identify a subset $S \subseteq B$ and a sparse vector $\mathbf{v}|_S \in \mathbb{C}^N$ satisfying*

$$\left\|\hat{\mathbf{f}} - \mathbf{v}|_S\right\|_2 \leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_2 + \frac{33}{\sqrt{s}} \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_1 + 198\sqrt{s}\,\|\mathbf{f}\|_\infty N^{-r}. \tag{5.8}$$

Algorithm 1's operation count is then

$$\mathcal{O}\left(\frac{s^2 \cdot r^{\frac{3}{2}} \cdot \log^{\frac{11}{2}}(N)}{\log(s)}\right).$$

If returning a sparse vector $\mathbf{v}|_S \in \mathbb{C}^N$ that satisfies (5.8) *with probability at least* $(1-p) \in [2/3, 1)$ *is sufficient, a Monte Carlo variant of the deterministic Algorithm 3 in [4] may be used in line 9 of Algorithm 1. In this case Algorithm 1's operation count is*

$$\mathcal{O}\left(s \cdot r^{\frac{3}{2}} \cdot \log^{\frac{9}{2}}(N) \cdot \log\left(\frac{N}{p}\right)\right).$$

Proof. Redefine δ in the proof of Theorem 7 in [4] as

$$\delta = \frac{1}{\tau}\left(\frac{1}{2s} \cdot \left\|\widehat{f} - \widehat{f}_s^{\text{opt}}\right\|_1 + 3\,\|\mathbf{f}\|_\infty N^{-r}\right) = 3\left(\frac{1}{2s} \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_1 + 3\,\|\mathbf{f}\|_\infty N^{-r}\right),$$

and observe that any $\omega \in B = \left[-\left\lceil \frac{N}{2} \right\rceil, \left\lfloor \frac{N}{2} \right\rfloor\right) \cap \mathbb{Z}$ that is reconstructed by Algorithm 1 will have a Fourier series coefficient estimate v_ω that satisfies

$$\left|v_\omega - \hat{\mathbf{f}}_\omega\right| = \left|v_\omega - \widehat{f}_\omega\right| \leq \sqrt{2} \cdot \delta.$$

We can thus bound the approximation error by

$$\begin{aligned}\left\|\hat{\mathbf{f}} - \mathbf{v}|_S\right\|_2 &\leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}|_S\right\|_2 + \left\|\hat{\mathbf{f}}|_S - \mathbf{v}|_S\right\|_2 \leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}|_S\right\|_2 + 2\sqrt{s}\cdot\delta \\ &= \sqrt{\left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_2^2 + \sum_{\omega \in R_s^{\text{opt}}\left(\widehat{f}\right)\setminus S} \left|\widehat{f}_\omega\right|^2 - \sum_{\tilde{\omega} \in S\setminus R_s^{\text{opt}}\left(\widehat{f}\right)} \left|\widehat{f}_{\tilde{\omega}}\right|^2} + 2\sqrt{s}\cdot\delta. \end{aligned} \tag{5.9}$$

In order to make additional progress on (5.9) we must consider the possible magnitudes of $\widehat{\mathbf{f}}$ entries at indices in $S\setminus R_s^{\text{opt}}\left(\widehat{f}\right)$ and $R_s^{\text{opt}}\left(\widehat{f}\right)\setminus S$. Careful analysis (in line with the techniques employed in the proof of Theorem 7 of [4]) indicates that

$$\sum_{\omega \in R_s^{\text{opt}}\left(\widehat{f}\right)\setminus S} \left|\widehat{f}_\omega\right|^2 - \sum_{\tilde{\omega} \in S\setminus R_s^{\text{opt}}\left(\widehat{f}\right)} \left|\widehat{f}_{\tilde{\omega}}\right|^2 \leq s \cdot \left(8\sqrt{2}+8\right)^2 \cdot \delta^2.$$

Therefore, in the worst possible case equation (5.9) will remain bounded by

$$\left\|\hat{\mathbf{f}} - \mathbf{v}|_S\right\|_2 \leq \sqrt{\left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_2^2 + s\cdot\left(8\sqrt{2}+8\right)^2\cdot\delta^2} + 2\sqrt{s}\cdot\delta \leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\text{opt}}\right\|_2 + 22\sqrt{s}\cdot\delta.$$

The error bound stated in (5.8) follows.

The runtimes follow by observing that $c_2 = \mathcal{O}\left(\alpha \cdot \log^{\frac{1}{2}}(N)\right) = \mathcal{O}\left(r^{\frac{1}{2}} \cdot \log^{\frac{1}{2}}(N)\right)$ as chosen in line 2 of Algorithm 1, and for every choise of q in line 4 of Algorithm 1, all of the evaluations $\left\{(\tilde{g}_q * f)(x_k)\right\}_{k=1}^{m}$ can be approximated very accurately in just $\mathcal{O}(mr\log N)$-time, where the number of samples m is on the orders described in Theorem 6. □

We are now ready to empirically evaluate Algorithm 1 with several different SFT algorithms $\mathcal{A}$ used in its line 9.

Chapter 6

Numerical Experiment

In this section we evaluate the performance of three new discrete SFT Algorithms resulting from Algorithm 1: DMSFT-4, DMSFT-6,[1] and CLW-DSFT.[2] All of them were developed by utilizing different SFT algorithms in line 9 of Algorithm 1. Here DMSFT stands for the **D**iscrete **M**ichigan **S**tate **F**ourier **T**ransform algorithm. Both DMSFT-4 and DMSFT-6 are implementations of Algorithm 1 that use a randomized version of the SFT algorithm GFFT [8] in their line 9.[3] The only difference between DMSFT-4 and DMSFT-6 is how accurately each one estimates the convolution in line 7 of Algorithm 1: for DMSFT-4 we use $\kappa = 4$ in the partial discrete convolution in lemma 11 when approximating $\tilde{g}_q * f$ at each x_k, while for DMSFT-6 we always use $\kappa = 6$. The CLW-DSFT stands for the **C**hristlieb **L**awlor **W**ang **D**iscrete **S**parse **F**ourier **T**ransform algorithm. It is an implementation of Algorithm 1 that uses the SFT developed in [6] in its line 9, and κ varying between 12 and 20 for its line 7 convolution estimates (depending on each input vector's Fourier sparsity, etc.). All of DMSFT-4, DMSFT-6 and CLW-DSFT were implemented in C++ in order to empirically evaluate their run time and noise robustness characteristics.

We also compare these new implementations' runtime and robustness characteristics

[1]The code for both DMSFT variants is available at https://sourceforge.net/projects/aafftannarborfa/.
[2]The CLW-DSFT code is available at www.math.msu.edu/ markiwen/Code.html.
[3]Code for GFFT is also available at www.math.msu.edu/ markiwen/Code.html.

with FFTW 3.3.4[4] and sFFT 2.0[5]. FFTW is the highly optimized FFT implementation which runs in $\mathcal{O}(N \log N)$-time for input vectors of length N. All the standard discrete Fourier Transforms in the numerical experiments are performed using FFTW 3.3.4 with FFTW_MEASURE plan. The sFFT 2.0 is a randomized discrete sparse Fourier Transform algorithm written in C++ which is both stable and robust to noise. It was developed by Indyk et al. in [23]. Note that DMSFT-4, DMSFT-6, CLW-DSFT, and sFFT 2.0 are all randomized algorithms designed to approximate discrete DFTs that are approximately s-sparse. This means that all of them take both sparsity s and size N of the DFT's $\hat{\mathbf{f}} \in \mathbb{C}^N$ they aim to recover as parameters. In contrast, FFTW can not utilize existing sparsity to its advantage. Finally, all experiments are run on a Linux CentOS machine with 2.50GHz CPU and 16 GB of RAM.

6.1 Experiment Setup

For the execution time experiments each trial input vector $\mathbf{f} \in \mathbb{C}^N$ was generated as follows: First s frequencies were independently selected uniformly at random from $[0, N) \cap \mathbb{Z}$, and then each of these frequencies was assigned a uniform random phase with magnitude 1 as its Fourier coefficient. The remaining frequencies' Fourier coefficients were then set to zero to form $\hat{\mathbf{f}} \in \mathbb{C}^N$. Finally, the trial input vector $\mathbf{f}$ was then formed via an inverse DFT.

For each pair of s and N the parameters in each randomized algorithm were chosen so that the probability of correctly recovering all s energetic frequencies was at least 0.9 per trial input. Every data point in a figure below corresponds to an average over 100 runs on 100 different trial input vectors of this kind. It is worth mentioning that the parameter

[4]This code is available at http://www.fftw.org/

[5]This code is available at https://groups.csail.mit.edu/netmit/sFFT/

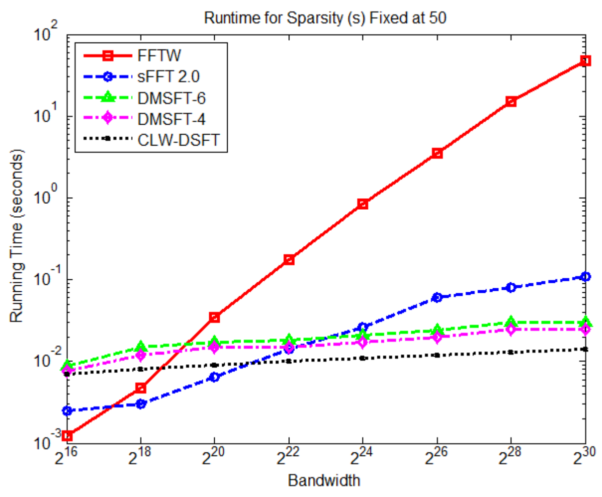


Figure 6.1: Runtime comparison at sparsity (s) Fixed at 50

tuning process for DMSFT-4 and DMSFT-6 requires significantly less effort than for both CLW-DSFT and sFFT 2.0 since the DMSFT variants only have two parameters (whose default values are generally near-optimal).

6.2 Runtime as Input Vector Size Varies

In Figure 6.1 we fixed the sparsity to $s = 50$ and ran numerical experiments on 8 different input vector lengths N: 2^{16}, 2^{18}, ..., 2^{30}. We then plotted the running time (averaged over 100 runs) for DMSFT-4, DMSFT-6, CLW-DSFT, sFFT 2.0, and FFTW.

As expected, the runtime slope of all the SFT algorithms (i.e. DMSFT-4, DMSFT-6, CLW-DSFT, and sFFT 2.0) is less than the slope of FFTW as N increases. Although FFTW is fastest for vectors of small size, it becomes the slowest algorithm when the vector size N is greater than 2^{20}. Among the randomized algorithms, sFFT 2.0 is the fastest one when N is less than 2^{22}, but DMSFT-4, DMSFT-6, and CLW-DSFT all outperform sFFT 2.0 with respect to runtime when the input vector's sizes are large enough. The CLW-DSFT implementation becomes faster than sFFT 2.0 when N is approximately 2^{21} while DMSFT-4

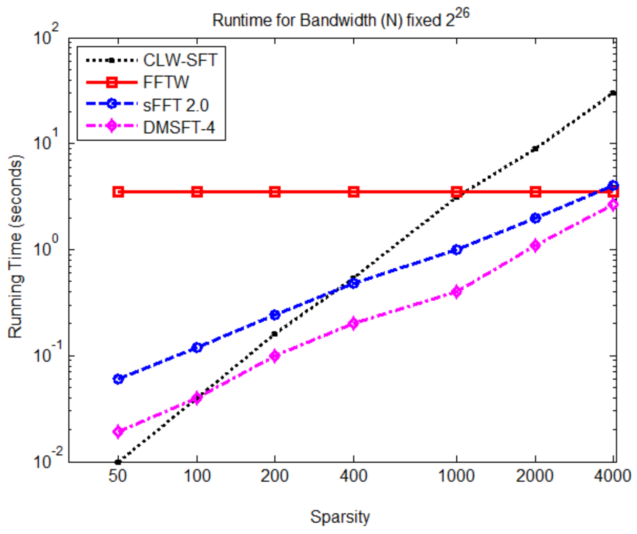


Figure 6.2: Runtime comparison at Bandwidth (N) Fixed at 2^{26}

and DMSFT-6 have better runtime performance than sFFT 2.0 when N is greater than 2^{23}.

6.3 Runtime as Sparsity Varies

In Figure 6.2 we fix the input vector lengths to $N = 2^{26}$ and run the numerical experiments on 7 different values of sparsity s: 50, 100, 200, 400, 1000, 2000, and 4000. As expected, the FFTW's runtime is constant as we increase the sparsity. The runtimes of DMSFT-4, CLW-DSFT, and sFFT 2.0 are all essentially linear in s. Here DMSFT-6 has been excluded for ease of viewing/reference – its runtimes lie directly above those of DMSFT-4 when included in the plot. Looking at Figure 2 we can see the CLW-DSFT's runtime increases more rapidly with s than that of DMSFT-4 and sFFT 2.0. The runtime of CLW-DSFT becomes the slowest one when sparsity is around 1000. DMSFT-4 and sFFT 2.0 have approximately the same runtime slope as s increases, and they both have good performance when the sparsity is large. However, DMSFT-4 maintains consistently better runtime performance than sFFT 2.0 for all sparsity values, and is the only algorithm in the plot that still faster than FFTW

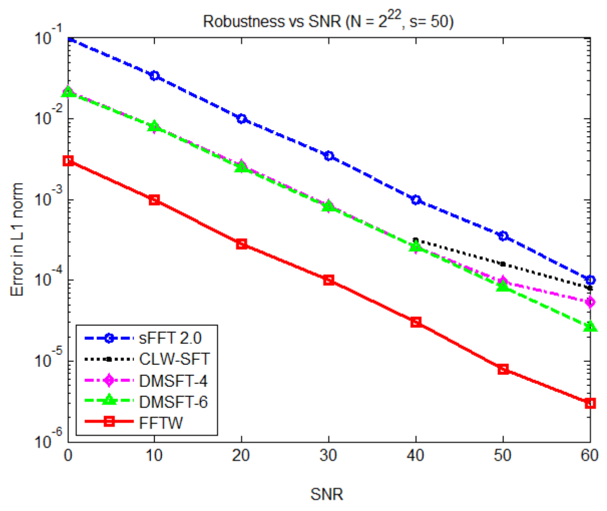


Figure 6.3: Robustness to Noise (bandwidth (N) = 2^{22}, sparsity (s) = 50).

when the sparsity is 4000. Indeed, when the sparsity is 4000 the average runtime of DMSFT-4 is 2.68s and the average runtime of DMSFT-6 is 2.9s. Both of them remain faster than FFTW (3.47s) and sFFT 2.0 (3.96s) for this large sparsity (though only DMSFT-4 has been included in the plot above).

6.4 Robustness to Noise

In our final set of experiments we test the noise robustness of DMSFT-4, DMSFT-6, CLW-DSFT, sFFT 2.0, and FFTW for different levels of Gaussian noise. Here the size of each input vector is $N = 2^{22}$ and sparsity is fixed at $s = 50$. The test signals are then generated as before, except that Gaussian noise is added to $\mathbf{f}$ after it is constructed. More specifically, we first generate $\mathbf{f}$ and then set $\mathbf{f} = \mathbf{f} + \mathbf{n}$ where each entry of $\mathbf{n}$, n_j, is an i.i.d. mean 0 random complex Gaussian value. The noise vector $\mathbf{n}$ is then rescaled to achieve each desired signal-to-noise ratio (SNR) considered in the experiments.[6]

[6]The SNR is defined as $SNR = 20 \log \frac{\|\mathbf{f}\|_2}{\|\mathbf{n}\|_2}$, where $\mathbf{f}$ is the length N input vector and $\mathbf{n}$ is the length N noise vector.

Recall that the the randomized algorithms compared herein (DMSFT-4, DMSFT-6, CLW-DSFT, and sFFT 2.0) are all tuned to guarantee exact recovery of s-sparse functions with probability at least 0.9 in all experiments. For our noise robustness experiments this ensures that the correct frequency support, S, is found for at least 90 of the 100 trial signals used to generate each point plotted in Figure 6.3. We use average L_1 error to measure the noise robustness of each algorithm for each of these at least 90 trial runs. The average L_1 error is defined as

$$Average\ L_1\ Error = \frac{1}{s}\sum_{\omega\in S}\left|\hat{f}_\omega - z_\omega\right|$$

where S is the true frequency support of the input vector $\mathbf{f}$, $\hat{f}_\omega$ are the true input Fourier coefficients for all frequencies $\omega \in S$, and z_ω are their recovered approximations from each algorithm. Figure 6.3 graphs the averaged average L_1 error over the at least 90 trial signals where each method correctly identified S.

It can be seen in Figure 6.3 that DMSFT-4, DMSFT-6, sFFT 2.0, and FFTW are all robust to noise. As expected, FFTW has the best performance in this test. DMSFT-4 and DMSFT-6 are both more robust to noise when compared to sFFT 2.0. As for CLW-DSFT, it cannot guarantee a 0.9 probability of correctly recovering S when the SNR is less than 40 and so is not plotted for those SNR values. This is due to the base energetic frequency identification methods of [10, 6] being inherently ill conditioned, though the CLW-DSFT results look better when compared to the true $\hat{\mathbf{f}}$ with respect to, e.g., earth mover's distance. Frequencies are often estimated incorrectly by CLW-DSFT at higher noise levels, but when they are they are usually at least close enough to the true frequencies to be informative.

Chapter 7

Sparse FFT for Functions with Structured Fourier Sparsity

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

$$f(x) = \sum_{j=1}^{n} \sum_{\omega \in S_j} c_\omega e^{\mathbb{i}\omega x} \tag{7.1}$$

where each unknown S_j has simplifying structure. (e.g., has $|x - y| < B \ll N$ for all $x, y \in S_j$). We say that the function in 7.1 is $(n, B)-$structured in Fourier space. From (7.1) one can see that each such f is approximately $Bn-$sparse. The Algorithm 1 in [30] is based on the best unstructured SFT algorithm in [4, 2]. As a result, it can recover any Bn-sparse function f in $B^2 n^2 \log^{\mathcal{O}(1)} N-$time. In this chapter, we focus on the results of the numerical experiments. We show that the Algorithm 1 in [30] is the best known deterministic SFT method for the recovery of polynomially structured sparse input functions by comparing its runtime and robustness characteristics with several other DFT/SFT algorithms. We point the reader to [30] for more theoretical details.

7.1 Numerical Experiments

In this section, whenever we mention the structured SFT algorithm we refer to the Algorithm 1 in [30]. Here we evaluate the performance of two different variants of the structured SFT algorithm including (*i*) the deterministic variant for block sparse functions described in section 4.2 of [30] (referred to as the **F**ourier **A**lgorithm for **S**tructured sparsi**T**y (FAST) below), and (*ii*) a randomized implementation of the structured SFT algorithm which only utilizes a small random subset of the hashing primes used by FAST for each choice of its parameters (referred to as the **F**ourier **A**lgorithm for **S**tructured sparsi**T**y with **R**andomization (FASTR) below). Both of these C++ implementations are publicly available [1]. We also compare these implementations' runtime and robustness with GFFT[2], CLW-SFT, FFTW 3.3.4[3] and sFFT 2.0[4].

Note that FAST and FASTR are both designed to approximate functions that are $(n, B)-$block sparse in Fourier space. This means that both FAST and FASTR take upper bounds on the number of blocks, n, and length of each block, B, present in the spectrum of the functions they aim to recover as parameters. In contrast, GFFT (a deterministic sparse Fourier transform [4]), CLW-SFT (a multiscale sub-linear time Fourier algorithm developed by Andrew Christlieb, David Lawlor and Yang Wang[6]) and SFFT 2.0 (a randomized noise robust sparse Fourier transform [23]) only require an upper bound on the effective sparsity,

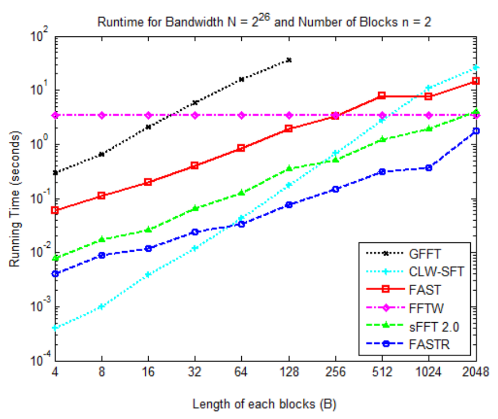
[1]http://na.math.uni-goettingen.de/index.php?section=gruppe&subsection=software
[2]http://users.math.msu.edu/users/markiwen/Code.html
[3]http://www.fftw.org/
[4]https://groups.csail.mit.edu/netmit/sFFT/

s, of the function's Fourier coefficients. Herein s is always set so that $s = Bn$ for these methods. Finally, FFTW is a highly optimized and publicly available implementation of the traditional FFT algorithm which runs in O(N log N)-time for input vectors of length N. All the FFTW results below were obtained using FFTW 3.3.4 with its FFTW_MEASURE plan.

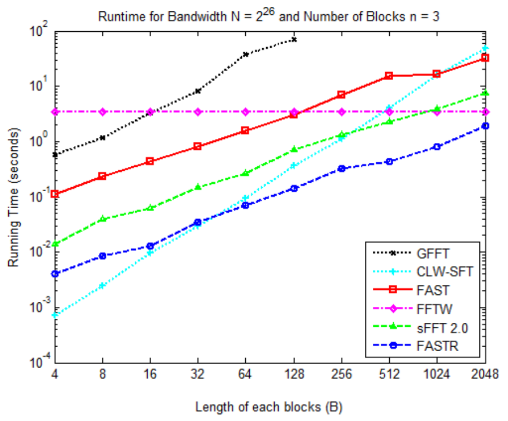
For the runtime experiments below the trial signals were formed by choosing sets of frequencies with $(n, B)-$block sparsity uniformly at random from $(-\lceil N/2 \rceil, \lfloor N/2 \rfloor] \cap \mathbb{Z}$. Each frequency in this set was then assigned a magnitude 1 Fourier coefficient with a uniformly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

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

In Figure 7.1a we fix the number of blocks to $n = 2$ and the bandwidth to $N = 2^{26}$, and then perform numerical experiments for 10 different block lengths $B = 2^2, 2^3, ..., 2^{11}$. We then plot the runtime (averaged over 100 trial runs) for FAST, FASTR, GFFT, CLW-SFT, sFFT 2.0 and FFTW. As expected, the runtime of FFTW is constant with increasing sparsity. The runtimes of all the sparse Fourier transform algorithms other than GFFT are approximately linear in B, and they have similar slopes. Figure 7.1a demonstrates that allowing a small probability of incorrect recovery always lets the randomized algorithms (FASTR and sFFT 2.0) outperform the deterministic algorithms with respect to runtime.



(a) Runtime comparison at bandwidth $N = 2^{26}$ and number of blocks $n = 2$



(b) Runtime comparison at bandwidth $N = 2^{26}$ and number of blocks $n = 3$

Figure 7.1: Runtime plots for several algorithms and implementations of sparse Fourier Transform under different setting

Among the deterministic algorithms, FAST is always faster than GFFT, and only becomes slower than FFTW when the value of B is greater than 256. The runtimes of both FASTR and sFFT 2.0 are still comparable with the one of FFTW when the block length B is 2048. Comparing with sFFT 2.0, FASTR has better runtime performance on these block sparse functions, and is the only algorithm that is still faster than FFTW when $B = 2048$. For CLW-SFT, it has best runtime performance when the sparsity is less than 64, however, its runtime increasing rapidly with the increasing of sparsity s and it becomes slower than FAST when the sparsity is 4096. In Figure 7.1b we use the same settings of N and B as in the previous experiment and increase the number of blocks n from 2 to 3. With these settings the largest sparsity $s = Bn$ increases from 4048 $(2 \cdot 2^{11})$ to 6144 $(3 \cdot 2^{11})$. The respective results for the methods are similar in this plot.

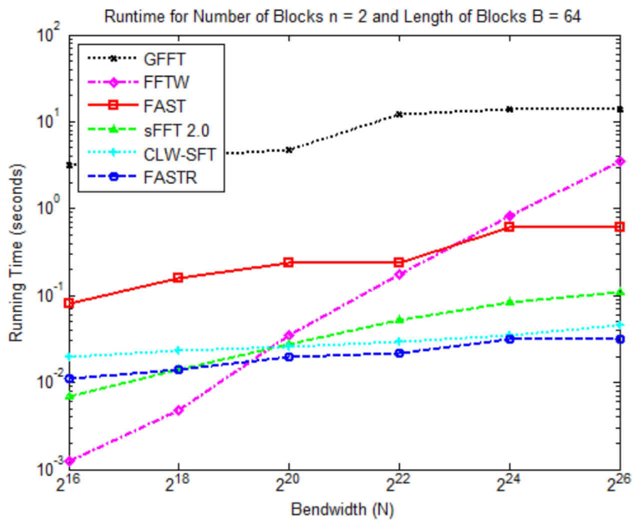


Figure 7.2: Runtime comparison for $n = 2$ blocks of length $B = 64$.

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

In Figure 7.2 we fix the number of blocks $n = 2$ and block length $B = 64$, then test the performance of the different algorithms with various bandwidths N. It can be seen in Figure 7.2 that FFTW is the fastest deterministic algorithm for small bandwidth values. However, the runtime of FFTW becomes slower than the one of FAST when the bandwidth N is greater than 2^{24}. GFFT is the slowest deterministic algorithm for this sparsity level for all plotted N. Comparing randomized SFT algorithms, FASTR always performs better than sFFT 2.0 when the bandwidth is greater than 2^{18}. The CLW-SFT has almost the same performance with FASTR, which is amazing since the CLW-SFT only needs sparsity information but does not have to know the structure of signal in Fourier space.

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

In Figure 7.3, we fix the bandwidth $N = 2^{26}$ and length of block $B = 32$, then vary the number of blocks n from 1 to 10. Looking at Figure 7.3 we can see the deterministic sparse

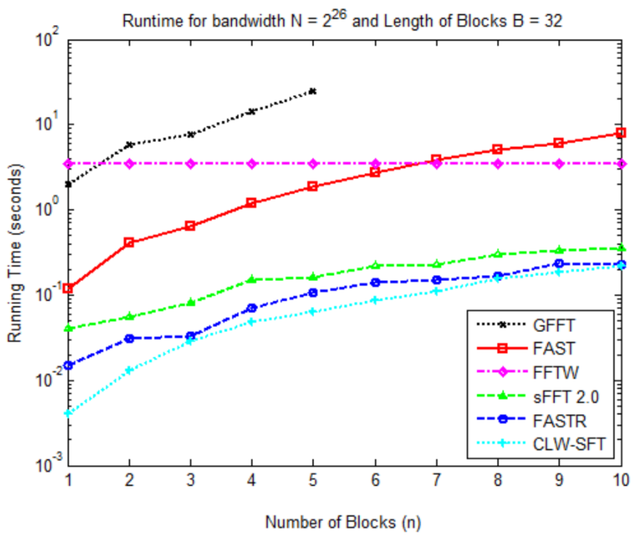


Figure 7.3: Runtime comparison for bandwidth $N = 2^{26}$ and block length $B = 32$.

FFTs, GFFT and FAST both have runtimes that increase more rapidly with n than those of their randomized competitors. Among the three deterministic algorithms, FAST has the best performance when the number of blocks is smaller than 6. Similar to the previous experiments, FFTW becomes the fastest deterministic algorithm when the sparsity $s = Bn$ gets large enough (greater than 224 in this experiment). The two randomized algorithms are both faster than FFTW by an order of magnitude when the number of blocks is 10. Similarly, FASTR is always faster than sFFT 2.0 for the examined value of N. Finally, the CLW-SFT has the best runtime performance in this numerical experiment. Actually, from Figure 7.3, Figure 7.1a, Figure 7.1b and Figure 7.2 we can see that the CLW-SFT almost always has the advantage (in the sense of runtime) when the sparsity s of the signal is small, e.g. couple hundreds. However, if the sparsity s is large and one has the prior knowledge of the structure of the signal in Fourier space, then FASTR is the best (randomized) algorithm comparing with all its competitors.

7.1.4 Robustness to Noise

To test the robustness of the methods to noise we add Gaussian noise to each of the signal samples utilized in each method and then measure the contamination of the recovered Fourier series coefficients for $(n, B)-$block sparse functions $f : [0, 2\pi] \to \mathbb{C}$ with bandwidth $N = 2^{22}$, number of blocks $n = 3$, and block length $B = 24$. More specifically, each method considered herein utilizes a set of samples from f given by $\mathbf{f} = (f(x_j))_{j=0}^{m-1}$ for some $x_0, ..., x_{m-1} \in [0, 2\pi)$ with $m \leq N$. For the experiments in this section we instead provide each algorithm with noisy function evaluations of the form $(f(x_j) + n_j)_{j=0}^{m-1}$, where each $n_j \in \mathbb{C}$ is a complex Gaussian random variable with mean 0. The n_j are then rescaled so that the total additive noise $\mathbf{n} = (n_j)_{j=0}^{m-1}$ achieves the signal-to-noise ratios (SNRs) considered in Figure 7.4. [5]

Recall that the two randomized algorithms compared herein (SFT 2.0 and FASTR) are both tuned to guarantee exact recover of block sparse functions with probability at least 0.9 in all experiments. For our noise robustness experiments this ensures that the correct frequency support, S, is found for at least 90 of the 100 trial signals used to generate each point plotted in Figure 7.4. All the other (deterministic) methods always find this correct support for all noise levels considered herein after sorting their output Fourier coefficient estimates by magnitude. Figure 7.4 plots the average ℓ_1- error over the true Fourier coefficients for frequencies in the correct frequency support S of each trial signal, averaged over the at least 90 trial runs at each point for which each sparse Fourier transform correctly identified S. More specifically, it graphs

$$\textit{Average } \ell_1 \textit{ Error} = \frac{1}{nB} \sum_{\omega \in S} \left| c_\omega - x_\omega \right| \tag{7.2}$$

[5]The SNR is defined as $SNR = 20 \log \frac{\|\mathbf{f}\|_2}{\|\mathbf{n}\|_2}$, where $\mathbf{f}$ and $\mathbf{n}$ are as given above.

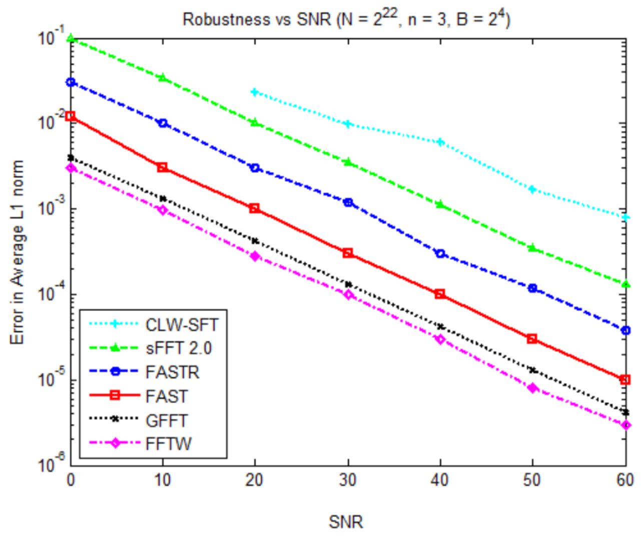


Figure 7.4: Robustness to Noise (bandwidth (N) = 2^{22}, sparsity (s) = 50).

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

Looking at Figure 7.4 one can see that all of the Fourier transform algorithms in our experiments are robust to noise. Overall, however, the deterministic algorithms (FAST, GFFT and FFTW) are more robust than randomized algorithms (FASTR and sFFT 2.0). As expected, FFTW is the most robust algorithm in this experiment, followed closely by GFFT. For the randomized algorithms, FASTR is more robust than sFFT 2.0. As for CLW-SFT, it cannot guarantee a 0.9 probability of correctly recovering S when the SNR is less than 20 and so is not plotted for those SNR values. This is because the CLW-SFT was not designed to guarantee to find the correct set of frequencies, but instead, the algorithm aim to find the frequency set that can explain the signal best. This makes the CLW-SFT results look better when compared to the true $\hat{\mathbf{f}}$ with respect to, e.g., earth mover's distance. Frequencies have a chance to be estimated incorrectly by CLW-SFT at higher noise levels, but when they are they are usually at least close enough to the true frequencies to be informative.

Chapter 8

Conclusion

This thesis contains two part. In the first part (from Chapter 3 to Chapter 6) we present a general method that can convert any SFT algorithm into a fully discrete SFT algorithm. More specifically, let $\mathcal{A}$ be a sublinear-time sparse FFT algorithm which utilizes unequally spaced samples from a given periodic function $f : [-\pi, \pi] \to \mathbb{C}$ in order to rapidly approximate its sequence of Fourier series coefficients $\hat{f} \in \ell^2$. In this thesis, we propose a generic method of transforming any such algorithm $\mathcal{A}$ into a sublinear-time sparse DFT algorithm which rapidly approximates $\hat{\mathbf{f}}$ from a given input vector $\mathbf{f} \in \mathbb{C}^N$. As a result, we can construct several new sublinear-time sparse DFT algorithms from existing sparse Fourier algorithms which utilize unequally spaced function samples [8, 4, 6, 10]. The best of these new algorithms is shown to outperform existing discrete sparse Fourier transform methods on both runtime and noise robustness for large vector lengths N. We also present several new theoretical discrete sparse FFT robust recovery guarantees. These include the first known theoretical guarantees for entirely deterministic and discrete sparse DFT algorithms which hold for arbitrary input vectors $\mathbf{f} \in \mathbb{C}^N$. It is worth to mention that the filter function and fast discrete convolution method described in Chapter 4 can be used in any applications involve rapid evaluation for the value of a convolution function. We want to emphasize here that though we choose to use the periodic Gaussian function as the filter function in this thesis, it is not the only possible choice. Actually, one of the possible future works is to find more filter functions that can be used in our frame. Besides that, one could also try to apply our method to higher dimensional algorithm [54], and structured SFT algorithm [30].

Then, in the second part (Chapter 7) we demonstrate the results of numerical experiments of the deterministic SFT method we developed in [30]. By comparing with several other DFT/SFT algorithms (FFTW, sFFT 2.0, GFFT), we show that the algorithm in [30] is the fastest known deterministic SFT method for the recovery of polynomially structured sparse input functions.

APPENDICES

Appendix A

Proof of Lemmas 3, 4 and 5

We will restate each lemma before its proof for ease of reference.

Lemma 16 (Restatement of Lemma 3)**.** *The $2\pi-$periodic Gaussian $g : [-\pi, \pi] \to \mathbb{R}^+$ has*

$$g(x) \leq \left(\frac{3}{c_1} + \frac{1}{\sqrt{2\pi}} \right) e^{-\frac{x^2}{2c_1^2}}$$

for all $x \in [-\pi, \pi]$.

Proof. Observe that

$$\begin{aligned} c_1 g(x) = \sum_{n=-\infty}^{\infty} e^{-\frac{(x-2n\pi)^2}{2c_1^2}} &= e^{-\frac{x^2}{2c_1^2}} + e^{-\frac{(x-2\pi)^2}{2c_1^2}} + e^{-\frac{(x+2\pi)^2}{2c_1^2}} + \sum_{|n|\geq 2} e^{-\frac{(x-2n\pi)^2}{2c_1^2}} \\ &\leq 3e^{-\frac{x^2}{2c_1^2}} + \int_1^{\infty} e^{-\frac{(x-2n\pi)^2}{2c_1^2}} dn + \int_1^{\infty} e^{-\frac{(x+2n\pi)^2}{2c_1^2}} dn \end{aligned}$$

holds since the series above have monotonically decreasing positive terms, and $x \in [-\pi, \pi]$.

Now, if $x \in [0, \pi]$ and $n \geq 1$, one has

$$e^{-\frac{(2n+1)^2\pi^2}{2c_1^2}} \leq e^{-\frac{(x+2n\pi)^2}{2c_1^2}} \leq e^{-\frac{4n^2\pi^2}{2c_1^2}} \leq e^{-\frac{(x-2n\pi)^2}{2c_1^2}} \leq e^{-\frac{(2n-1)^2\pi^2}{2c_1^2}},$$

which yields

$$\begin{aligned}
c_1 g(x) &\leq 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + 2\int_1^{\infty} \mathrm{e}^{-\frac{\pi^2(2n-1)^2}{2c_1^2}}\, dn \\
&= 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{1}{2}\left(\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{\pi^2 m^2}{2c_1^2}}\, dm - \int_{-1}^{1} \mathrm{e}^{-\frac{\pi^2 m^2}{2c_1^2}}\, dm\right) \\
&= 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{c_1}{\sqrt{2\pi}} - \frac{1}{2}\int_{-1}^{1} \mathrm{e}^{-\frac{\pi^2 m^2}{2c_1^2}}\, dm.
\end{aligned}$$

Using lemma 9 to bound the last integral we can now get that

$$\begin{aligned}
c_1 g(x) &\leq 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{c_1}{\sqrt{2\pi}} - \frac{1}{2}\frac{\sqrt{2}c_1}{\pi}\sqrt{\pi\left(1 - \mathrm{e}^{-\frac{\pi^2}{2c_1^2}}\right)} \\
&= 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{c_1}{\sqrt{2\pi}}\left(1 - \sqrt{\left(1 - \mathrm{e}^{-\frac{\pi^2}{2c_1^2}}\right)}\right) \\
&\leq 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{c_1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\pi^2}{2c_1^2}} \\
&\leq 3\mathrm{e}^{-\frac{x^2}{2c_1^2}} + \frac{c_1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2c_1^2}}.
\end{aligned}$$

Recalling now that g is even we can see that this inequality will also hold for all $x \in [-\pi, 0]$ as well. □

Lemma 17 (Restatement of Lemma 4)**.** *The 2π−periodic Gaussian $g : [-\pi, \pi] \to \mathbb{R}^+$ has*

$$\widehat{g}_\omega = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{c_1^2\omega^2}{2}}$$

for all $\omega \in \mathbb{Z}$. Thus, $\widehat{g} = \{\widehat{g}_\omega\}_{\omega\in\mathbb{Z}} \in \ell^2$ decreases monotonically as $|\omega|$ increases, and also *has* $\|\widehat{g}\|_\infty = \frac{1}{\sqrt{2\pi}}$.

Proof. Starting with the definition of the Fourier transform, we calculate

$$\begin{aligned}
\widehat{g}_\omega &= \frac{1}{c_1}\sum_{n=-\infty}^{\infty}\frac{1}{2\pi}\int_{-\pi}^{\pi} \mathbb{e}^{-\frac{(x-2n\pi)^2}{2c_1^2}}\, \mathbb{e}^{-\mathbb{i}\omega x}\, dx \\
&= \frac{1}{c_1}\sum_{n=-\infty}^{\infty}\frac{1}{2\pi}\int_{-\pi}^{\pi} \mathbb{e}^{-\frac{(x-2n\pi)^2}{2c_1^2}}\, \mathbb{e}^{-\mathbb{i}\omega(x-2n\pi)}\, dx \\
&= \frac{1}{c_1}\sum_{n=-\infty}^{\infty}\frac{1}{2\pi}\int_{-\pi-2n\pi}^{\pi-2n\pi} \mathbb{e}^{-\frac{u^2}{2c_1^2}}\, \mathbb{e}^{-\mathbb{i}\omega u}\, du \\
&= \frac{1}{2\pi c_1}\int_{-\infty}^{\infty} \mathbb{e}^{-\frac{u^2}{2c_1^2}}\, \mathbb{e}^{-\mathbb{i}\omega u}\, du \\
&= \frac{c_1\sqrt{2\pi}}{2\pi c_1}\mathbb{e}^{-\frac{c_1^2\omega^2}{2}} \\
&= \frac{\mathbb{e}^{-\frac{c_1^2\omega^2}{2}}}{\sqrt{2\pi}}.
\end{aligned}$$

The last two assertions now follow easily. ☐

Lemma 18 (Restatement of Lemma 5)**.** *Choose any* $\tau \in \left(0, \frac{1}{\sqrt{2\pi}}\right)$, $\alpha \in \left[1, \frac{N}{\sqrt{\ln N}}\right]$, *and* $\beta \in \left(0, \alpha\sqrt{\frac{\ln(1/\tau\sqrt{2\pi})}{2}}\,\right]$. *Let* $c_1 = \frac{\beta\sqrt{\ln N}}{N}$ *in the definition of the periodic Gaussian* g *from* (4.3)*. Then* $\widehat{g}_\omega \in \left[\tau, \frac{1}{\sqrt{2\pi}}\right]$ *for all* $\omega \in \mathbb{Z}$ *with* $|\omega| \leq \left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil$.

Proof. By Lemma 4 above it suffices to show that

$$\frac{1}{\sqrt{2\pi}}\mathbb{e}^{-\frac{c_1^2\left(\left\lceil \frac{N}{\alpha\sqrt{\ln N}}\right\rceil\right)^2}{2}} \geq \tau,$$

which holds if and only if

$$\begin{aligned} c_1^2 \left(\left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil \right)^2 &\leq 2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right) \\ c_1 &\leq \frac{\sqrt{2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right)}}{\left\lceil \frac{N}{\alpha\sqrt{\ln N}} \right\rceil}. \end{aligned}$$

Thus, it is enough to have

$$c_1 \leq \frac{\sqrt{2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right)}}{\frac{N}{\alpha\sqrt{\ln N}}+1} = \frac{\alpha\sqrt{2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right)\ln N}}{N+\alpha\sqrt{\ln N}},$$

or,

$$c_1 = \frac{\beta\sqrt{\ln N}}{N} \leq \frac{\alpha\sqrt{2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right)\ln N}}{2N} \leq \frac{\alpha\sqrt{2\ln\left(\frac{1}{\tau\sqrt{2\pi}}\right)\ln N}}{N+\alpha\sqrt{\ln N}}.$$

This, in turn, is guaranteed by our choice of β. $\square$

Appendix B

Proof of Lemma 12 and Theorem 6

We will restate lemma 12 before it's proof for ease of reference.

Lemma 19 (Restatement of Lemma 12)**.** *Let $s, \epsilon^{-1} \in \mathbb{N} \setminus \{1\}$ with $(s/\epsilon) \geq 2$, and $\mathbf{n} \in \mathbb{C}^m$ be an arbitrary noise vector. There exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ such that Algorithm 3 on page 72 of [4], when given access to the corrupted samples $\{f(x_k) + n_k\}_{k=1}^m$, will identify a subset $S \subseteq B$ which is guaranteed to contain all $\omega \in B$ with*

$$\left|\widehat{f}_\omega\right| > 4\left(\frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty\right). \tag{B.1}$$

Furthermore, every $\omega \in S$ returned by Algorithm 3 will also have an associate Fourier series coefficient estimate $z_\omega \in \mathbb{C}$ which is guaranteed to have

$$\left|\widehat{f}_\omega - z_\omega\right| \leq \sqrt{2}\left(\frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty\right). \tag{B.2}$$

Both the number of required samples, m, and Algorithm 3's operation count are

$$\mathcal{O}\left(\frac{s^2 \cdot \log^4(N)}{\log\left(\frac{s}{\epsilon}\right) \cdot \epsilon^2}\right). \tag{B.3}$$

If succeeding with probability $(1 - \delta) \in [2/3, 1)$ is sufficient, and $(s/\epsilon) \geq 2$, the Monte Carlo variant of Algorithm 3 referred to by Corollary 4 on page 74 of [4] may be used. This

Monte Carlo variant reads only a randomly chosen subset of the noisy samples utilized by the deterministic algorithm,

$$\{f(\tilde{x}_k) + \tilde{n}_k\}_{k=1}^{\tilde{m}} \subseteq \{f(x_k) + n_k\}_{k=1}^{m},$$

yet it still outputs a subset $S \subseteq B$ *which is guaranteed to simultaneously satisfy both of the following properties with probability at least* $1 - \delta$*:*

(i) S *will contain all* $\omega \in B$ *satisfying* (B.1), *and*

(ii) all $\omega \in S$ *will have an associated coefficient estimate* $z_\omega \in \mathbb{C}$ *satisfying* (B.2).

Finally, both this Monte Carlo variant's number of required samples, $\tilde{m}$*, as well as its operation count will also always be*

$$\mathcal{O}\left(\frac{s}{\epsilon} \cdot \log^3(N) \cdot \log\left(\frac{N}{\delta}\right)\right). \tag{B.4}$$

Proof. The proof of this lemma involves a somewhat tedious and uninspired series of minor modifications to various results from [4]. In what follows we will outline the portions of that paper which need to be changed in order to obtain the stated lemma. Algorithm 3 on page 72 of [4] will provide the basis of our discussion.

In the first paragraph of our lemma we are provided with m-contaminated evaluations of f, $\{f(x_k) + n_k\}_{k=1}^{m}$, at the set of m points $\{x_k\}_{k=1}^{m} \subset [-\pi, \pi]$ required by line 4 of Algorithm 1 on page 67 of [4]. These contaminated evaluations of f will then be used to approximate the vector $\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A} \in \mathbb{C}^m$ in line 4 of Algorithm 3. More specifically, using (18) on page 67 of [4] one can see that each $\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j \in \mathbb{C}$ is effectively computed via a DFT

$$\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j = \frac{1}{s_j}\sum_{k=0}^{s_j-1} f\left(-\pi + \frac{2\pi k}{s_j}\right)\mathbb{e}^{\frac{-2\pi \mathbb{i} k h_j}{s_j}} \tag{B.5}$$

for some integers $0 \leq h_j < s_j$. Note that we are guaranteed to have noisy evaluations of f at each of these points by assumption. That is, we have $f\left(x_{j,k}\right) + n_{j,k}$ for all $x_{j,k} := -\pi + \frac{2\pi k}{s_j}$, $k = 0, \ldots, s_j - 1$.

We therefore approximate each $\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j$ via an approximate DFT as per (B.5) by

$$E_j := \frac{1}{s_j}\sum_{k=0}^{s_j-1}\left(f\left(x_{j,k}\right) + n_{j,k}\right)\mathbb{e}^{\frac{-2\pi \mathbb{i} k h_j}{s_j}}.$$

One can now see that

$$\left|E_j - \left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j\right| = \left|\frac{1}{s_j}\sum_{k=0}^{s_j-1} n_{j,k}\mathbb{e}^{\frac{-2\pi \mathbb{i} k h_j}{s_j}}\right| \leq \frac{1}{s_j}\sum_{k=0}^{s_j-1}\left|n_{j,k}\right| \leq \|\mathbf{n}\|_\infty \tag{B.6}$$

holds for all j. Every entry of both $\mathcal{E}_{s_1,K}\tilde{\psi}\mathbf{A}$ and $\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}$ referred to in Algorithm 3 will therefore be effectively replaced by its corresponding E_j estimate. Thus, the lemma we seek to prove is essentially obtained by simply incorporating the additional error estimate (B.6) into the analysis of Algorithm 3 in [4] wherever an $\mathcal{E}_{s_1,K}\tilde{\psi}\mathbf{A}$ or $\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}$ currently appears.

To show that lines 6 – 14 of Algorithm 3 will identify all $\omega \in B$ satisfying (B.1) we can adapt the proof of Lemma 6 on page 72 of [4]. Choose any $\omega \in B$ you like. Lemmas 3 and

5 from [4] together with (B.6) above ensure that both

$$\begin{aligned}\left|E_j - \widehat{f}_\omega\right| &\leq \left|E_j - \left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j\right| + \left|\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j - \widehat{f}_\omega\right| \\ &\leq \|\mathbf{n}\|_\infty + \frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 \end{aligned} \tag{B.7}$$

and

$$\begin{aligned}\left|E_{j'} - \widehat{f}_\omega\right| &\leq \left|E_{j'} - \left(\mathcal{E}_{s_1,K}\tilde{\psi}\mathbf{A}\right)_{j'}\right| + \left|\left(\mathcal{E}_{s_1,K}\tilde{\psi}\mathbf{A}\right)_{j'} - \widehat{f}_\omega\right| \\ &\leq \|\mathbf{n}\|_\infty + \frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 \end{aligned} \tag{B.8}$$

hold for more than half of the j and j'-indexes that Algorithm 3 uses to approximate $\widehat{f}_\omega$. The rest of the proof of Lemma 6 now follows exactly as in [4] after the δ at the top of page 73 is redefined to be $\delta := \frac{\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}^{\text{opt}}_{(s/\epsilon)}\right\|_1}{s} + \left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty$, each $\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j$ entry is replaced by E_j, and each $\left(\mathcal{E}_{s_1,K}\tilde{\psi}\mathbf{A}\right)_{j'}$ entry is replaced by $E_{j'}$.

Similarly, to show that lines 15 – 18 of Algorithm 3 will produce an estimate $z_\omega \in \mathbb{C}$ satisfying (B.2) for every $\omega \in S$ one can simply modify the first few lines of the proof of Theorem 7 in Appendix F of [4]. In particular, one can redefine δ as above, replace the appearance of each $\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j$ entry by E_j, and then use (B.7). The bounds on the runtime follow from the last paragraph of the proof of Theorem 7 in Appendix F of [4] with no required changes. To finish, we note that the second paragraph of the lemma above follows from a completely analogous modification of the proof of Corollary 4 in Appendix G of [4].

□

Now we are ready to give the prove of Theorem 4.

Theorem 6 (Restatement of Theorem 4)**.** *Suppose $f : [-\pi, \pi] \to \mathbb{C}$ has $\widehat{f} \in \ell^1 \cap \ell^2$. Let $s, \epsilon^{-1} \in \mathbb{N} \setminus \{1\}$ with $(s/\epsilon) \geq 2$, and $\mathbf{n} \in \mathbb{C}^m$ be an arbitrary noise vector. Then, there exists a set of m points $\{x_k\}_{k=1}^m \subset [-\pi, \pi]$ together with a simple deterministic algorithm $\mathcal{A} : \mathbb{C}^m \to \mathbb{C}^{4s}$ such that $\mathcal{A}\left(\{f(x_k) + n_k\}_{k=1}^m\right)$ is always guaranteed to output (the nonzero coefficients of) a degree $\leq N/2$ trigonometric polynomial $y_s : [-\pi, \pi] \to \mathbb{C}$ satisfying*

$$\|f - y_s\|_2 \leq \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_s^{\mathrm{opt}}\right\|_2 + \frac{22\epsilon \cdot \left\|\hat{\mathbf{f}} - \hat{\mathbf{f}}_{(s/\epsilon)}^{\mathrm{opt}}\right\|_1}{\sqrt{s}} + 22\sqrt{s}\left(\left\|\widehat{f} - \widehat{f}|_B\right\|_1 + \|\mathbf{n}\|_\infty\right). \tag{B.9}$$

Both the number of required samples, m, and the algorithm's operation count are always

$$\mathcal{O}\left(\frac{s^2 \cdot \log^4(N)}{\log\left(\frac{s}{\epsilon}\right) \cdot \epsilon^2}\right). \tag{B.10}$$

If succeeding with probability $(1 - \delta) \in [2/3, 1)$ is sufficient, and $(s/\epsilon) \geq 2$, a Monte Carlo variant of the deterministic algorithm may be used. This Monte Carlo variant reads only a randomly chosen subset of the noisy samples utilized by the deterministic algorithm,

$$\{f(\tilde{x}_k) + \tilde{n}_k\}_{k=1}^{\tilde{m}} \subseteq \{f(x_k) + n_k\}_{k=1}^m,$$

yet it still outputs (the nonzero coefficients of) a degree $\leq N/2$ trigonometric polynomial, $y_s : [-\pi, \pi] \to \mathbb{C}$, that satisfies (B.9) *with probability at least $1 - \delta$. Both its number of required samples, $\tilde{m}$, as well as its operation count will always be*

$$\mathcal{O}\left(\frac{s}{\epsilon} \cdot \log^3(N) \cdot \log\left(\frac{N}{\delta}\right)\right). \tag{B.11}$$

Proof. To get the first paragraph of Theorem 6 one can simply utilize the proof of Theorem 7 exactly as it is written in Appendix F of [4] after redefining δ as above, and then replacing the appearance of each $\left(\mathcal{G}_{\lambda,K}\tilde{\psi}\mathbf{A}\right)_j$ entry with its approximation E_j. Once this has been done, equation (42) in the proof of Theorem 7 can then be taken as a consequence of lemma 12 above. In addition, all references to lemma 6 of [4] in the proof can then also be replaced with appeals to lemma 12 above. To finish, the proof of Corollary 4 in Appendix G of [4] can now be modified in a completely analogous fashion in order to prove the second paragraph of Theorem 6. $\square$

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. A. Iwen. Compressed Sensing with Sparse Binary Matrices: Instance Optimal Error Guarantees in Near-Optimal Time. *Journal of Complexity, Vol. 30, Issue 1, pages 1 – 15, 2014.*

[2] M. A. Iwen. Combinatorial Sublinear-Time Fourier Algorithms. *Foundations of Computational Mathematics, Vol. 10, Issue 3, pages 303 – 338, 2010.*

[3] J. Bailey, M. A. Iwen, and C. V. Spencer. On the Design of Deterministic Matrices for Fast Recovery of Fourier Compressible Functions. *SIAM J. Matrix Anal. Appl., Vol. 33, No. 1, pages 263 – 289, 2012.*

[4] M. A. Iwen. Improved Approximation Guarantees for Sublinear-Time Fourier Algorithms. *Applied and Computational Harmonic Analysis, Vol. 34, Issue 1, pages 57 – 82, 2013.*

[5] M. A. Iwen. Additional notes on Lemma 6.
`http://users.math.msu.edu/users/markiwen/Papers/Lemma6_FOCM_10.pdf`

[6] Andrew Christlieb, David Lawlor, and Yang Wang. A Multiscale Sub-linear Time Fourier Algorithm for Noisy Data. *Submitted, 2013. arXiv:1304.4517*

[7] Gerald B. Folland. Fourier Analysis and Its Applications. *Wadsworth& Brooks/Cole Mathematics Series*

[8] I.B. Segal, M.A. Iwen. Improved Sparse Fourier Approximation Results: Faster Implementations and Stronger Guarantees. *Numerical Algorithms, Vol. 63, Issue 2, pages 239 – 263, 2013.*

[9] Anna C. Gilbert, Piotr Indyk, Mark Iwen. Recent Developments in the Sparse Fourier Transform. *GlobalSIP, 2013.*

[10] D. Lawlor, Y. Wang, and A. Christlieb. Adaptive sub-linear time Fourier algorithms. *Advances in Adaptive Data Analysis, vol. 5, no. 1, 2013.*

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

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

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

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

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

[16] D. L. Donoho and P. B. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math. 49 (1989), no. 3, 906-931.*

[17] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp. 19 (1965), 297-301.*

[18] Charles Van Load. Computational Frameworks for the Fast Fourier Transform. *SIAM, 1992*

[19] Dongarra, J., Sullivan, F. Guest Editors Introduction to the top 10 algorithms. *Computing in Science Engineering. 2 (1): 22-23. doi:10.1109/MCISE.2000.814652. ISSN 1521-9615.*

[20] Strang, Gilbert. Wavelets. *American Scientist. 82 (3): 253. JSTOR 29775194.*

[21] Shannon, Claude E. A Mathematical Theory of Communication. *Bell System Technical Journal. 27 (3): 379-423. doi:10.1002/j.1538-7305.1948.tb01338.x.*

[22] M. Iwen, A. Gilbert, and M. Strauss. Empirical evaluation of a sub-linear time sparse DFT algorithm. *Commun. Math. Sci. 5 (2007), no. 4, 981-998*

[23] Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price. Simple and practical algorithms for sparse Fourier transform. *ACM-SIAM Symposium on Discrete Algorithms (SODA), 2012.*

[24] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *IEEE 93 (2005), no. 2, 216-231, Special issue on Program Generation, Optimization, and Platform Adaptation.*

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

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

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

[28] J. Schumacher. High Performance sparse fast fourier transform. *Master thesis, Computer Science, ETH Zurich, Switzerland, 2013.*

[29] C. Wang, M. Araya-Polo, S. Chandrasekaran, A. St-Cyr, B. Chapman, and D. Hohl. Parallel sparse FFT. *SC Workshop on Irregular Applications: Architectures and Algorithms, 2013.*

[30] Sina Bittens, Ruochuan Zhang, Mark A. Iwen. A Deterministic Sparse FFT for Functions with Structured Fourier Sparsity. *https://arxiv.org/abs/1705.05256*

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

[32] O. Abari, E. Hamed, H. Hassanieh, A. Agarwal, D. Katabi, A. P. Chandrakasan, and V. Stojanovic. A 0.75-million-point Fourier-transform chip for frequencysparse signals. *ISSCC, 2014.*

[33] M. Iwen. A deterministic sub-linear time sparse Fourier algorithm via non-adaptive compressed sensing methods. *ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, 2008.*

[34] M. Ajtai, H. Iwaniec, J. Komlos, J. Pintz, and E. Szemeredi. Construction of a thin set with small Fourier coefficients. *Bull. London Math. Soc. 22 (1990), no. 6, 583-590.*

[35] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *Annual Symposium on Foundations of Computer Science, vol. 44, 2003, pp. 146-159.*

[36] D. Boneh. Finding smooth integers in short intervals using crt decoding. *Journal of Computer and System Sciences 64 (2002), no. 4, 768-784.*

[37] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *J. AMS 22 (2009), no. 1, 211-231.*

[38] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput. 14 (1993), no. 6, 1368-1393.*

[39] A. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier representations via sampling. *Symposium on Theory of Computing, 2002, pp. 152-161.*

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

[41] A. Gilbert, M. Strauss, and J. Tropp. Near-optimal sparse A tutorial on fast Fourier sampling. *IEEE Signal Processing Magazine 25 (2008), no. 2, 57-66.*

[42] O. Goldreich, D. Ron, and M. Sudan. Near-optimal sparse Chinese remaindering with errors. *IEEE Transactions on Information Theory 46 (2000), no. 4, 1330-1338.*

[43] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse fourier transform. *to appear in ACM Symposium on Theory of Computing (STOC), 2012.*

[44] M. Iwen, A. Gilbert, and M. Strauss. Nearly optimal sparse fourier transform DFT algorithm. *Commun. Math. Sci. 5 (2007), no. 4, 981-998.*

[45] N. Katz. An estimate for character sums. *J. Amer. Math. Soc. 2 (1989), no. 2, 197-200.*

[46] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. Comput. 22 (1993), no. 6, 1331-1348.*

[47] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *J. Assoc. Comput. Mach. 40 (1993), no. 3, 607-620.*

[48] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Computing 24 (1995), no. 2, 357-368.*

[49] I.E. Shparlinski and R. Steinfeld. Noisy chinese remaindering in the Lee norm. *Journal of Complexity 20 (2004), no. 2-3, 423-437.*

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

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

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

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

[54] Bosu Choi, Andrew Christlieb, Yang Wang. Multi-dimensional Sublinear Sparse Fourier Algorithm. *https://arxiv.org/abs/1606.07407*

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