SUB-LINEAR FOURIER ALGORITHMS: THEORY AND IMPLEMENTATION

By

David Lawlor

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Applied Mathematics

2012

ABSTRACT

SUB-LINEAR FOURIER ALGORITHMS: THEORY AND IMPLEMENTATION

By

David Lawlor

We present a new deterministic algorithm for the sparse Fourier transform problem, in which we seek to identify $k \ll N$ significant Fourier coefficients from a signal of bandwidth N. Previous deterministic algorithms exhibit quadratic runtime scaling in k, while our algorithm scales linearly with k in the average case. Underlying our algorithm are a few simple observations relating the Fourier coefficients of time-shifted samples to unshifted samples of the input function. This allows us to detect when aliasing between two or more frequencies has occurred, as well as to determine the value of unaliased frequencies. We also show that the small time shifts required for our algorithm can be implemented using two much larger shifts, an important point for applications where the sampling rate is limited by hardware capacity. Empirically our algorithm is orders of magnitude faster than competing algorithms.

We study the effect of noise on the performance of our algorithm, and derive a relationship between its magnitude and the rate at which the signal is sampled. This in turn is used to devise modified versions of our basic algorithm that are robust to noise, although with higher runtime and sampling requirements than in the noiseless case. The first of these is a minor change, and exhibits poor performance relative to the noise level. To remedy this we propose a multi-scale algorithm that allows for error correction in the frequency estimates in an iterative fashion. It is shown empirically that this multi-scale algorithm is robust to noise even with coarse sampling rates.

Finally, we discuss the application of our algorithms to the multi-dimensional setting. A straightforward extension, while simple to describe and implement, exhibits exponential scaling with the dimension. A second extension is then proposed that uses number-theoretic techniques to reduce the multi-dimensional problem to the one-dimensional case, which can then be solved efficiently with the algorithm proposed in chapter two. An empirical evaluation shows that this second method significantly outperforms highly optimized FFT implementations in two dimensions.

To my family

ACKNOWLEDGMENTS

I would first like to thank my advisor, Andrew Christlieb, for his advice and guidance over the past five years. In our very first meeting he showed the patience and wisdom necessary to mold my vague notions of research problems into definite plans. He made sure I kept on pace throughout my graduate career, which sometimes meant pushing me outside of my comfort zone. Beyond his scientific guidance, his advice on navigating an academic career has been invaluable.

I would like to thank Anna Gilbert for the many hours spent working through ideas in her office, acting at times as a proxy advisor for me in Ann Arbor. She generously invited me to attend seminars, group meetings, and workshops at the University of Michigan and elsewhere that have helped to define my research interests. I would like to thank Yang Wang for providing the spark that would become my dissertation project, and for finding time in his busy schedule as chair of our department to work through problems and provide feedback on drafts. I would like to thank Jianliang Qian and Di Liu for serving on my defense committee and for ably teaching the core applied mathematics courses during my graduate studies.

I would like to thank the support staff of the math department, and Barbara Miller in particular, for all their help with administrative matters. I would like to thank my fellow math students for their friendship over the past five years. The nights spent playing softball, sharing dinner, or laughing over beers made the experience of graduate school more rewarding. The applied math group deserves special mention for organizing a student seminar where I was able to practice giving research talks in front of a friendly audience.

My friends from college and outside the math department have given me many fond memories having nothing to do with academics. Many thanks go to them for helping me to

not miss out on life while pursuing my studies. My parents, Jack and Marian, my brother Patrick, and my grandfather Bill have been incredibly supportive throughout my graduate career, and I dedicate this dissertation to them. Lastly I would like to express my profound gratitude to my partner Melanie, who has shown me the deepest love, kindness, and patience throughout our time together. It is no exaggeration to say that this dissertation would not have been possible without her support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Sparse Fourier approximation

The Fast Fourier Transform (FFT) [12] is arguably the most ubiquitous numerical algorithm in scientific computing. In addition to being named one of the "Top Ten Algorithms" of the past century [15], the FFT is a critical tool in myriad applications, ranging from signal processing to computational PDE and machine learning. At the time of its introduction, it represented a major leap forward in the size of problems that could be solved on available hardware, as it reduces the runtime complexity of computing the Discrete Fourier Transform (DFT) of a length-N array from $\mathrm{O}(N^2)$ to $\mathrm{O}(N \log N)$. Even with advanced hardware and optimized implementations, for large N this still represents the major computational bottleneck in many applications.

Any algorithm which computes all N Fourier coefficients necessarily has a runtime complexity of $\Omega(N)$, since it takes that much time merely to report the output. However, in many applications it is known that the DFT of the signal of interest is highly sparse – that is,

only a small number of coefficients are non-zero. In this case it is possible to break the $\Omega(N)$ barrier by asking only for the largest k terms in the signal's DFT. When $k \ll N$ existing algorithms for this *sparse Fourier approximation* problem can significantly outperform even highly optimized FFT implementations [30, 28, 26].

Any sub-linear[1] Fourier algorithm can't even *read* all the input, and so must either be approximate or succeed with high probability (or both). The discrete uncertainty principle of Donoho and Stark [16] provides a heuristic justification for sub-sampling in the time domain if the signal of interest is highly sparse. One version states that if N_t, N_ω are the number of non-zero values of a length-N signal respectively in the time and frequency domains, then it must hold that $N_t N_\omega \geq N$. Thus when $N_\omega \ll N$, the number of non-zero time samples must be large. This suggests that the few samples taken by a sub-linear algorithm should still carry significant information about the frequency content of the signal.

To date, most sparse Fourier approximation algorithms have exhibited an empirical speedup over the FFT only for limited sparsity regimes due to their complicated structure, and so are sub-linear only in an asymptotic sense.[2] In contrast, the algorithms presented in this dissertation have a simple structure which permits a straightforward, fast implementation, resulting in improvements over the FFT for a large range of sparsity values.

[1] That is, an algorithm with runtime $o(N)$.

[2] The recent algorithm [26] has been shown to outperform the FFT for a much larger range of k, comparable to that of the noiseless algorithm presented in Chapter 2 of this dissertation.

1.2 Relationship to compressed sensing

The term "compressed sensing" refers to a new paradigm in signal processing which seeks to recover a compressible signal from a number of linear measurements roughly proportional to its information content, rather than its nominal dimension. There has been tremendous recent progress in this area, as evidenced by the daily updates of a popular research blog devoted to the subject [9]. While this dissertation does not make explicit use of the results or algorithms of compressed sensing, there are parallels in the approaches used. The purpose of this section is to clarify the relationship between the two.

All algorithms for the sparse approximate DFT problem take a small number of samples of the input x, either at random or in a deterministic fashion. These samples are then processed in a highly non-linear, algorithm-dependent manner to produce a k-term Fourier representation of x – that is, a list $\left\{\left(\widetilde{\omega}_\ell, \widetilde{a}_\ell\right)\right\}_{\ell=1}^{k}$ of significant frequency/coefficient pairs. In other words, these algorithms approximately solve the severely underdetermined system $\boldsymbol{R}\boldsymbol{F}^*\widehat{x} = \boldsymbol{R}x$, where $\boldsymbol{R}$ is the restriction to the samples used by the algorithm, $\boldsymbol{F}^*$ is the adjoint of the $N \times N$ discrete Fourier matrix with entries

$$F_{jk} = \frac{1}{\sqrt{N}} \mathrm{e}^{-2\pi \mathrm{i} jk/N}, \ 0 \leq j, k < N, \tag{1.1}$$

and $\widehat{x}$ is the discrete Fourier transform of x.

In [8], Candès, Romberg, and Tao considered the dual problem: that of recovering a given signal from highly incomplete Fourier measurements. Specifically, suppose that a signal x of

length N is the superposition of k spikes at times $t = \tau_j$:

$$x[t] = \sum_{j=1}^{k} x[\tau_j]\delta(t - \tau_j). \tag{1.2}$$

The authors show that, with high probability, x can be recovered exactly from a randomly chosen set Ω of m frequencies from the discrete Fourier transform of x, provided

$$m \geq Ck \log N \tag{1.3}$$

for some constant C whose value depends on the desired probability of success. This can be viewed as the severely underdetermined linear system dual to the system described above: $\boldsymbol{RF}x = \boldsymbol{R}\widehat{x}$. The recovery algorithm in this case is the ℓ_1 minimization

$$g^* = \operatorname{argmin} \|g\|_1 \quad \text{subject to } \widehat{g}(\omega) = \widehat{f}(\omega) \text{ for all } \omega \in \Omega. \tag{1.4}$$

The idea of using ℓ_1 minimization to recover sparse vectors has been studied extensively in a number of research communities, including seismic imaging [42], image processing [41], and signal processing [10], where it is commonly referred to as basis pursuit. The theoretical foundations of ℓ_1 approximation are treated in depth in the monograph [39].

Sparse Fourier approximation and compressed sensing are therefore broadly similar in both their goals (sparse approximation of signals) and methods (in particular, the use of randomization.) There are, however, substantial differences between the two, which we now enumerate.

1. Sampling requirements. The compressed sensing model requires measurement matrices

to satisfy the Restricted Isometry Property, which has been shown to hold with high probability for random Gaussian, Bernoulli, and Fourier ensembles. Sparse Fourier algorithms, on the other hand, generally require more structure in their sampling sets. This is obviously true for the deterministic algorithms, and also for some of the randomized versions – in particular, [21] requires samples that lie on arithmetic progressions.

2. Reconstruction costs. As mentioned above, the reconstruction of the target signal in the compressed sensing model is achieved by a convex optimization problem (which can be recast as a linear program.) This is expected to incur a computational costs of $\mathrm{O}(N^3)$ for a signal of length N. Most Sparse Fourier approximation algorithms are *sublinear* in time, and so exponentially faster.

3. Allocation of resources. The balance between the two previous items is the major point of distinction. Indeed, we view the comparison of the two paradigms as an "apples-to-oranges" scenario: In the seismic imaging environment (where practitioners have recently implemented compressed sensing methods [35], [14]), high acquisition costs make long processing times on the back end more palatable. Sparse Fourier approximation algorithms, however, were developed with data streaming applications in mind. In this arena, low signal acquisition costs and enormous problem sizes necessitate fast algorithms with sparing use of memory resources.

1.3 Related work

The first works to implicitly address the sparse approximate DFT problem appeared in the theoretical computer science literature in the early 1990s. In [36], a variant of the Fourier transform for Boolean functions was shown to have applications for learnability. A polynomial-time algorithm to find large coefficients in this basis was given in [33], while the interpolation of sparse polynomials over finite fields was considered in [37]. It was later realized [21] that this last algorithm could be considered as an approximate DFT for the special case when N is a power of two.

In the past ten or so years, a number of algorithms have appeared which directly address the problem of computing sparse approximate Fourier transforms. When comparing the results in the literature, care must be taken to identify the class of signals over which a specific algorithm is to perform, as well as to identify the error bounds of a given method. Different algorithms have been devised in different research communities, and so have varying assumptions on the underlying signals as well as different levels of acceptable error.

The first result with sub-linear runtime and sampling requirements appeared in [20]. They give a $\mathrm{poly}(k, \log N, \log(1/\delta), 1/\varepsilon)$ time algorithm for finding, with probability $1 - \delta$, an approximation $\widehat{y}$ of the DFT of the input $\widehat{x}$ that is nearly optimal, in the sense that

$$\|\widehat{x} - \widehat{y}\|_2^2 \leq (1 + \varepsilon) \left\|\widehat{x} - \widehat{x}_{\mathrm{opt}}\right\|_2^2, \tag{1.5}$$

where $\widehat{x}_{\mathrm{opt}}$ is the best k-term approximation to $\widehat{x}$. This type of error bound is known as an "ℓ_2/ℓ_2" guarantee. Here the exponent of k in the runtime is two, so the algorithm is *quadratic* in the sparsity. Moreover, the algorithm is non-adaptive in the sense that the

samples used are independent of the input x. This algorithm was modified in [21] to bring the dependence on k down to linear.[3] This was accomplished mainly by replacing uniform random variables (used to sample the input) by random arithmetic progressions, which allowed the use of the nonequispaced fast Fourier transform to sample from intermediate representations and to estimate the coefficients in near-linear time. The increased overhead of this procedure, however, limited the range of k for which the algorithm outperformed a standard FFT implementation [30].

Around the same time, a similar algorithm was developed in the context of list decoding for proving hard-core predicates for one-way functions [3]. This can be considered an extension of [33], and like [20, 21] is a randomized algorithm. Since the goal in this work was to give a polynomial-time algorithm for list decoding, no effort was made to optimize the dependence on k; it stands at $k^{11/2}$, considerably higher than [20, 21]. The randomness in this algorithm is used only to construct a sample set on which norms are estimated, and in [2] this set is replaced with a deterministic construction. This construction is based on the notion of ε-approximating the uniform distribution over arithmetic progressions, and relies on existing constructions of ε-biased sets of small size [32, 1]. Depending on the size of the ε-biased sets used, the sampling and runtime complexities are $\mathrm{O}(k^4 \log^c N)$ and $\mathrm{O}(k^6 \log^c N)$, respectively, for some $c > 4$.[4]

[3] See [22] for a "user-friendly" description of the improved algorithm.

[4] Specifically, the runtime is $\mathrm{O}(k^2 \cdot \log N \cdot |S|)$, where S is the set of samples read by the algorithm. This set takes the form $S = \bigcup_{\ell=1}^{\lfloor \log N \rfloor} A - B_\ell$, where A has ε-discrepancy on rank 2 Bohr sets, B_ℓ ε-approximates the uniform distribution on $[0, 2^\ell) \cap \mathbb{Z}$, and $A - B_\ell$ is the difference set. Using constructions from [32] one has $|A| = \mathrm{O}(\varepsilon^{-1} \log^4 N)$, $|B_\ell| = \mathrm{O}(\varepsilon^{-3} \log^4 N)$; setting $\varepsilon = \Theta(k^{-1})$ and noting that $\left|\bigcup A - B_\ell\right| = \mathrm{O}\left(\sum \left|A - B_\ell\right|\right)$ and $\left|A - B_\ell\right| = \mathrm{O}(|A||B_\ell|)$ (see, e.g., [45]) one obtains the stated sampling and runtime complexities.

In the series of works [27, 28, 29], a different deterministic algorithm for sparse Fourier approximation was given that relies on the combinatorial properties of *aliasing*, or collisions among frequencies in sub-sampled DFTs. By taking enough short DFTs of co-prime lengths, and employing the Chinese Remainder Theorem to reconstruct energetic frequencies from their residues modulo these sample lengths, the author is able to prove sampling and runtime bounds of $\mathrm{O}(k^2 \log^4 N)$. The error bound is of the form

$$\|\widehat{x} - \widehat{y}\|_2 \leq \left\|\widehat{x} - \widehat{x}_{\mathrm{opt}}\right\|_2 + \frac{1}{\sqrt{k}} \left\|\widehat{x} - \widehat{x}_{\mathrm{opt}}\right\|_1 ; \tag{1.6}$$

it has been shown that the stronger "ℓ_2/ℓ_2" guarantee (1.5) cannot hold for a sub-linear, deterministic algorithm [11]. Moreover, the range of k for which this algorithm is faster than the FFT is smaller in practice than that of [21].

Most recently, the authors of [26] presented a randomized algorithm that extends by an order of magnitude the range of sparsity for which it is faster than the FFT. This is accomplished by removing the iterative aspect from [21] by using more efficient filters, which are nearly flat within the passband and which decay exponentially outside. In contrast, the box-car filters used in [21] have a frequency response which oscillates and decays like $|\omega|^{-1}$. In addition, the identification of significant frequencies is done by direct estimation after hashing into a large number of bins rather than the binary search technique of [21]. These changes give a runtime bound of $\mathrm{O}(\log N \sqrt{Nk \log N})$ and a somewhat stronger error bound

$$\|\widehat{x} - \widehat{y}\|_\infty^2 \leq \frac{\varepsilon}{k} \left\|\widehat{x} - \widehat{x}_{\mathrm{opt}}\right\|_2^2 + \delta \|\widehat{x}\|_1^2 \tag{1.7}$$

with probability $1 - 1/N$, where $\varepsilon > 0$ and $\delta = N^{-\mathrm{O}(1)}$ is a precision parameter.

These existing algorithms generally take one of two approaches to the sparse Fourier transform problem. In [20, 3, 21, 26], the spectrum of the input is randomly permuted and then run through a low-pass filter to isolate and identify frequencies which carry a large fraction of the signal's energy. This leads to randomized algorithms that fail on a non-negligible set of possible inputs. On the other hand, [28] takes advantage of the combinatorial properties of *aliasing* in order to identify the significant frequencies. This leads to a deterministic algorithm with higher runtime and sampling requirements than the randomized algorithms mentioned. Both of these randomized and deterministic approaches have drawbacks. Randomized algorithms are not suitable for failure-intolerant applications, while the process used to reconstruct significant frequencies in [28] relies on the Chinese Remainder Theorem (CRT), which is highly unstable to errors in the residues. While there do exist algorithms for "noisy Chinese Remaindering" [23, 5, 44] these have thus far not found application to the sparse DFT problem, and we leave this as future work.

As this dissertation was being prepared, the author became aware of an independent work using very similar methods for frequency estimation in the noiseless case [25]. Both methods consider the phase difference between Fourier samples to extract frequency information, but are based on different techniques for binning significant frequencies. The authors of [25] use random dilations and efficient filters of [26], whereas we use different sample lengths in the spirit of [28]. We believe both contributions are of interest, and reinforce the notion that exploiting phase information is critical for developing fast, robust algorithms for sparse Fourier approximation.

1.4 Thesis outline

The remainder of this dissertation is structured as follows. In Chapter 2 we give an average-case $\Theta(k \log(k))$ algorithm for the sparse Fourier transform in the noiseless setting. The runtime bound is proven over a certain class of random signals, and it is verified empirically with numerical simulations. We also show how to implement the algorithm using lower-resolution hardware, an important consideration for applications.

In Chapter 3 we extend the algorithm given in Chapter 2 to handle noisy signals. We first give a minor modification of the algorithm in chapter 2 to handle low-level noise, and we discuss the merits of measuring the error of approximation using the earth-mover distance (EMD) in place of the standard Euclidean norm. We then give two algorithms to handle more general noise. This is done in a multi-scale manner, where we learn chunks of significant frequencies in an iterative fashion.

Finally, in Chapter 4 we present extensions of our algorithm to multiple dimensions. Since the runtime of the FFT is $\mathrm{O}(N^d \log N)$ in d dimensions, we expect to see the greatest improvement over traditional Fourier transform methods in this setting. The first extension is quite natural and follows from the separability of the Fourier exponentials $\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\omega} \cdot \boldsymbol{x}}$ in multiple dimensions, but has runtime and sampling complexities that scale exponentially in the dimension d. We also give a second extension which relies on a clever application of the Chinese Remainder Theorem to construct an equivalent one-dimensional problem that is solved using the methods of chapter 2. This second algorithm significantly outperforms both the previous extension and highly-optimized FFT implementations, as shown empirically in two dimensions.

Chapter 2

Adaptive sub-linear Fourier algorithms

2.1 Introduction

In this chapter we describe a simple, deterministic algorithm that avoids reconstruction with the Chinese Remainder Theorem. As mentioned in chapter , existing deterministic algorithms for the sparse Fourier approximation problem rely on the CRT for energetic frequency reconstruction, and so are very unstable to errors in the residues. Our method relies on sampling the signal in the time domain at slightly shifted points, and thus it assumes access to an underlying continuous-time signal. The shifted time samples allow us to determine the value of significant frequencies in sub-sampled FFTs and also indicate when two or more frequencies have been aliased in such a sub-sampled FFT. These two key facts allow us to significantly reduce (by up to two orders of magnitude) the average-case sampling and runtime complexity of the sparse FFT over a certain class of random signals.

Our worst-case bounds improve by a constant factor those of prior deterministic algorithms. We present both adaptive and non-adaptive versions of our algorithms. If the application allows samples to be acquired adaptively (that is, dependent on previous samples), we are able to improve further on our average-case bounds.

The remainder of this chapter is organized as follows. In section 2.2 we introduce notation and prove the technical lemmas underlying our algorithms. In section 2.3 we introduce randomized and deterministic versions of our algorithm. In section 2.4 we prove that our algorithm has average-case runtime and sampling complexities of $\Theta(k \log(k))$ and $\Theta(k)$, respectively. In section 2.5 we present the results of an empirical evaluation of our algorithm and compare its runtime and sampling requirements to competing algorithms. In section 2.6 we describe how to combine two or more large shifts to reduce the sampling rate requirements of the algorithm, making them more suitable for hardware implementation. Finally in section 2.7 we provide some concluding remarks.

2.2 Mathematical background

In this section we lay the groundwork for the remainder of this chapter, and indeed the rest of this dissertation. In section 2.2.1 we explain the notation used in the sequel and give some elementary results concerning the relationship between Fourier series and the discrete Fourier transform. We also derive two key results concerning the phase of DFT coefficients and the relative magnitude of aliased coefficients that are at the heart of our algorithms. In section 2.2.2 we prove two lemmas which will be useful in deriving tests for aliasing and worst-case bounds for our algorithms.

2.2.1 Notation and preliminaries

Throughout this dissertation we shall be concerned with frequency-sparse band-limited signals $S : [0, 1) \to \mathbb{C}$ of the form

$$S(t) = \sum_{j=1}^{k} a_j \mathrm{e}^{2\pi \mathrm{i} \omega_j t}, \tag{2.1}$$

where $\omega_j \in [-N/2, N/2) \cap \mathbb{Z}$, $a_j \in \mathbb{C}$, and $k \ll N$. The Fourier series of S is given by

$$\widehat{S}(\omega) = \int_0^1 S(t) \mathrm{e}^{-2\pi \mathrm{i} \omega t} \mathrm{d}t, \ \omega \in \mathbb{Z}. \tag{2.2}$$

For signals of the form (2.1) we have

$$\begin{aligned} \widehat{S}(\omega_\ell) &= \int_0^1 \left(\sum_{j=1}^{k} a_j \mathrm{e}^{2\pi \mathrm{i} \omega_j t} \right) \mathrm{e}^{-2\pi \mathrm{i} \omega_\ell t} \mathrm{d}t \\ &= \sum_{j=1}^{k} a_j \int_0^1 \mathrm{e}^{2\pi \mathrm{i} (\omega_j - \omega_\ell) t} \mathrm{d}t. \end{aligned} \tag{2.3}$$

Since ω_j and ω_ℓ are integers, the integral in (2.3) is simply the Kronecker delta function $\delta_{j\ell}$, so we have $\widehat{S}(\omega_\ell) = a_\ell$ for $\ell = 1, \ldots, k$ and $\widehat{S}(\omega) = 0$ for all other $\omega \in [-N/2, N/2) \cap \mathbb{Z}$.

We use the "big-oh" asymptotic notation common in computer science, which we summarize here briefly. For two functions $f, g : \mathbb{N} \to \mathbb{R}^+$ we say $f = \mathrm{O}(g)$ if there is a positive constant C so that for all n large enough we have $f(n) \leq Cg(n)$. We say $f = \Theta(g)$ if $f = \mathrm{O}(g)$ and $g = \mathrm{O}(f)$. Finally, we say $f = \mathrm{o}(g)$ if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$.

We use the notation $\mathbb{P}[A]$ to denote the probability of an event A, and the notation $\mathbb{E}[X]$ to denote the expected value of a random variable X. The underlying probability measure

will always be clear from context.

Given any finite sequence $\boldsymbol{S} = (s_0, s_1, \ldots, s_{p-1})$ of length p we define its discrete Fourier transform (DFT) by

$$\widehat{\boldsymbol{S}}[h] \;=\; \sum_{j=0}^{p-1} s_j \mathrm{e}^{2\pi \mathrm{i} jh/p} \;=\; \sum_{j=0}^{p-1} \boldsymbol{S}[j] W_p^{jh}, \tag{2.4}$$

where $h = 0, 1, \ldots, p-1$, $\boldsymbol{S}[j] := s_j$ and $W_p := \mathrm{e}^{-2\pi \mathrm{i}/p}$ is the primitive p-th root of unity. For a detailed discussion of the DFT and its relationship to Fourier series see [6]. The Fast Fourier Transform (FFT) [12] allows the computation of $\widehat{\boldsymbol{S}}$ in $\mathrm{O}(p \log p)$ time for any integer p. See [46] for an in-depth treatment of the FFT and its many possible implementations.

The simplest way to determine the non-zero Fourier coefficients of a signal of the form (2.1) is to sample at the Nyquist rate $1/N$ and compute the full DFT. While guaranteed to give the correct output, this procedure has sampling and runtime complexities of $\Theta(N)$ and $\Theta(N \log N)$, respectively. This is computationally wasteful, since the assumption $k \ll N$ means that almost all coefficients of the Nyquist-rate DFT are zero. We are thus interested in *sub-linear* algorithms, for which both the runtime and sampling complexities are $\mathrm{o}(N)$. A frequency-sparse signal with k non-zero Fourier coefficients can be described using only $\Theta(k)$ space, since we need only to store the locations of the non-zero Fourier coefficients and the values of those coefficients. In section 2.3 we use the notation

$$\boldsymbol{R} \stackrel{\text{def}}{=} \left\{ \left(\omega_j, a_j \right) \right\}_{j=1}^{k} \tag{2.5}$$

to denote such a representation.

We apply the DFT to discrete samples of $S(t)$ to compute the Fourier coefficients a_j of

$S(t)$. For an integer p and real $\varepsilon > 0$ we form two discrete arrays $\boldsymbol{S}_p$ and $\boldsymbol{S}_{p,\varepsilon}$ by sampling $S(t)$ at rate $1/p$ starting at $t = 0$ and $t = \varepsilon$, respectively. That is, for $j = 0, 1, \ldots, p-1$,

$$\boldsymbol{S}_p[j] = S\Big(\frac{j}{p}\Big), \tag{2.6}$$

$$\boldsymbol{S}_{p,\varepsilon}[j] = S\Big(\frac{j}{p} + \varepsilon\Big). \tag{2.7}$$

If $S(t)$ is of the form (2.1), we have for the unshifted samples that

$$\begin{aligned}
\widehat{\boldsymbol{S}}_p[h] &= \sum_{j=0}^{p-1} \boldsymbol{S}_p[j] \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \sum_{j=0}^{p-1} S\left(\frac{j}{p}\right) \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \sum_{j=0}^{p-1} \left(\sum_{\ell=1}^{k} a_\ell \mathrm{e}^{2\pi \mathrm{i} \omega_\ell j/p} \right) \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \sum_{\ell=1}^{k} a_\ell \left(\sum_{j=0}^{p-1} \mathrm{e}^{2\pi \mathrm{i} j(\omega_\ell - h)/p} \right) \\
&= p \sum_{\omega_\ell \equiv h \ (\mathrm{mod}\ p)} a_\ell .
\end{aligned} \tag{2.8}$$

Here we used the fact that $\displaystyle\sum_{j=0}^{p-1} \mathrm{e}^{2\pi \mathrm{i} j\alpha}$ equals p if $\alpha \in \mathbb{Z}$ and zero otherwise in the last equality. For the shifted samples we have, similarly,

$$\begin{aligned}
\widehat{\boldsymbol{S}}_{p,\varepsilon}[h] &= \sum_{j=0}^{p-1} \boldsymbol{S}_{p,\varepsilon}[j] \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \sum_{j=0}^{p-1} S\left(\frac{j}{p} + \varepsilon\right) \mathrm{e}^{-2\pi \mathrm{i} hj/p}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=0}^{p-1} \left(\sum_{\ell=1}^{k} a_\ell \mathrm{e}^{2\pi \mathrm{i} \omega_\ell (j/p+\varepsilon)} \right) \mathrm{e}^{-2\pi \mathrm{i} h j/p} \\
&= \sum_{\ell=1}^{k} a_\ell \mathrm{e}^{2\pi \mathrm{i} \omega_\ell \varepsilon} \left(\sum_{j=0}^{p-1} \mathrm{e}^{2\pi \mathrm{i} j(\omega_\ell - h)/p} \right) \\
&= p \sum_{\omega_\ell \equiv h \pmod{p}} a_\ell \mathrm{e}^{2\pi \mathrm{i} \omega_\ell \varepsilon}. \qquad (2.9)
\end{aligned}
$$

Now assume that all $\omega_j \pmod{p},\ 1 \le j \le k$ are distinct. In this case we have from (2.8) that

$$
\widehat{\boldsymbol{S}}_p[h] = \begin{cases} p a_j & h = \omega_j \pmod{p} \\ 0 & \text{otherwise.} \end{cases}
$$

By examining the peaks of $\boldsymbol{S}_p$ we can determine $\omega_j \pmod{p}$ for $1 \le j \le k$. Futhermore, from (2.9) we have

$$
\widehat{\boldsymbol{S}}_{p,\varepsilon}[h] = \begin{cases} p a_j e^{2\pi \mathrm{i} \varepsilon \omega_j} & h = \omega_j \pmod{p} \\ 0 & \text{otherwise.} \end{cases}
$$

Using the both the ε-shifted and the unshifted samples, we can recover the frequencies $\omega_j \pmod{\varepsilon^{-1}}$ by noting that, for $h = \omega_j \pmod{p}$, we have $\dfrac{\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}_p[h]} = \mathrm{e}^{2\pi \mathrm{i} \varepsilon \omega_j}$. Hence

$$
2\pi\varepsilon\omega_j \equiv \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}_p[h]} \right) \pmod{2\pi}, \qquad (2.10)
$$

where $\operatorname{Arg}(z)$ denotes the phase angle of the complex number z in $[-\pi, \pi)$. Assume that we take $\varepsilon \le \frac{1}{N}$. Then ω_j is completely determined by (2.10) as there will be no wrap-around

aliasing, and we have

$$\omega_j = \frac{1}{2\pi\varepsilon} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}_p[h]}\right). \tag{2.11}$$

Remark 2.2.1. In fact, more generally, if we have an estimate of ω_j, say $|\omega_j| < \frac{L}{2}$, then by taking $\varepsilon \leq \frac{1}{L}$ the same reconstruction formula (2.11) holds; we will use this observation in chapter when we develop a multi-scale version of our algorithm for the case of noisy input. The fact that taking slightly shifted samples allows us to identify frequencies in $S(t)$ underlies the algorithms which follow, and the remainder of this chapter analyzes various aspects of the proposed algorithms, such as efficiency and robustness.

Equation (2.8) represents the phenomenon of *aliasing*, which occurs when two or more frequencies present in $S(t)$ are congruent modulo the sampling rate p. This can happen whenever $p < N$, and in this case it is impossible to distinguish the aliased frequencies using only samples at $t = j/p$. When aliasing occurs reconstruction via (2.11) is no longer valid. The aliasing phenomenon presents a serious challenge for any method with sub-linear sampling complexity. In the next section we develop a simple test to determine whether or not aliasing has occurred in a p-length DFT, which then allows us to effectively overcome this challenge and develop provably correct sub-linear algorithms.

2.2.2 Technical lemmas

To effectively apply the sub-sampling idea in a Fourier algorithm one must first overcome the aliasing challenge. Previous approaches applied the Chinese Remainder Theorem to reconstruct the frequencies ω_j by taking a suitable number of p's, which must overcome the problem of registrations to match up the residues whenever a new p is used. The complexity of this matching problem becomes combinatorial in k, and clever techniques which require

more samples from $S(t)$ must be used to maintain sub-linearity; see [28, 29] for details.

Using shifted sub-samples gives us a simple yet extremely effective criterion to determine whether or not aliasing has occurred at a given location in a p-length DFT without resorting to complicated combinatorial techniques. Denote by $I(S, h; p)$ the set of frequencies present in $S(t)$ that are congruent to $h \pmod p$, so that $I(S, h; p) \stackrel{\text{def}}{=} \{j : \omega_j \equiv h \pmod p\}$. From equations (2.8) and (2.9) we have

$$\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right|^2 - \left|\widehat{\boldsymbol{S}}_{p}[h]\right|^2 = p^2 \sum_{j,\ell \in I(S,h;p)} a_j \overline{a_\ell} \mathrm{e}^{2\pi \mathrm{i} \varepsilon(\omega_j - \omega_\ell)} - p^2 \Big| \sum_{j \in I(S,h;p)} a_j \Big|^2. \tag{2.12}$$

Lemma 2.2.2. *Let $p > 1$ and $h \in \{0, 1, \ldots, p-1\}$. Assume that $q = |I(S, h; p)| > 1$, i.e. $\omega_j \equiv h \pmod p$ for more than one j in $S(t)$. Then we have the following:*

(A) *Let $\varepsilon > 0$ and $E := \{\omega_j - \omega_m : j, m \in I(S, h; p)\}$. Suppose that all elements of εE are distinct $\pmod 1$. Then $\left|\widehat{\boldsymbol{S}}_{p,m\varepsilon}[h]\right| \neq \left|\widehat{\boldsymbol{S}}_{p}[h]\right|$ for some $1 \leq m \leq q^2 - q$.*

(B) *For almost all $\varepsilon > 0$ we have $\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right| \neq \left|\widehat{\boldsymbol{S}}_{p}[h]\right|$.*

Proof. The proof of part (B) is immediate from (2.12). Observe that $f(\varepsilon) := \left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right|^2 - \left|\widehat{\boldsymbol{S}}_{p}[h]\right|^2$ is trigonometric polynomial in ε, and it is not identically 0 given that $q = |I(S, h; p)| > 1$. Thus it has at most finitely many zeros for $\varepsilon \in [0, 1)$, and hence (B) is clearly true.

We resort to the Vandermonde matrix to prove part (A). Recall that a Vandermonde matrix V satisfies $V_{ij} = x_i^{j-1}$ for some values x_i. It is well-known that if the x_i are distinct then V is invertible [24]. For simplicity we write $f(t) = \sum_{\alpha \in E} c_\alpha \mathrm{e}^{2\pi \mathrm{i} \alpha t}$. Set $r_\alpha := \mathrm{e}^{2\pi \mathrm{i} \alpha \varepsilon}$ where ε satisfies the hypothesis of the lemma, which implies that all r_j are distinct. Assume

the claim of part (A) is false. Then we have $f(m\varepsilon) = 0$ for all $0 \leq m \leq q^2 - q$. Here $f(0) = 0$ is automatic because $\boldsymbol{S}_{p,0} = \boldsymbol{S}_p$. Thus we have

$$\sum_{\alpha \in E} c_\alpha r_\alpha^m = 0, \quad m = 0, 1, \ldots, q^2 - q. \tag{2.13}$$

But the cardinality of E is at most $q^2 - q + 1$, which means that there are at most $q^2 - q + 1$ terms in the sum in (2.13). By hypothesis all r_α are distinct, so the matrix $[r_\alpha^m]$ is a nonsingular Vandermonde matrix. Thus for (2.13) to hold all c_α must be zero. This is clearly not the case, and a contradiction. □

Remark 2.2.3. Any irrational ε or $\varepsilon = \frac{a}{b}$ with a, b coprime and $b \geq 2N$ will satisfy the hypothesis of part (A) of Lemma 2.2.2. For irrational ε, suppose to the contrary that there were a quadruplet (j_1, j_2, j_3, j_4) of indices so that

$$\varepsilon \left(\omega_{j_1} - \omega_{j_2} \right) \equiv \varepsilon \left(\omega_{j_3} - \omega_{j_4} \right) \pmod 1. \tag{2.14}$$

Then we would have

$$\varepsilon \left(\omega_{j_1} - \omega_{j_2} - \omega_{j_3} + \omega_{j_4} \right) \in \mathbb{Z}, \tag{2.15}$$

and since all ω_{j_i} are integers this would imply $\varepsilon \in \mathbb{Q}$, a contradiction. Similarly, in the case when $\varepsilon = \frac{a}{b}$ in lowest terms with $b \geq 2N$, suppose there were a quadruplet of indices so that (2.14) held. Since each $\left| \omega_{j_i} \right| \leq \frac{N}{2}$, the term in parentheses in (2.15) is less than $2N$. Thus we would have

$$n = \frac{a}{b} \left(\omega_{j_1} - \omega_{j_2} - \omega_{j_3} + \omega_{j_4} \right)$$

for some $n \in \mathbb{Z}$, but this is impossible since by assumption $b \geq 2N$.

Remark 2.2.4. It is also easy to show that in the special case where all coefficients a_j are real and $|I(S,h;p)| = 2$, we have $\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right| \neq \left|\widehat{\boldsymbol{S}}_{p}[h]\right|$ for any $\varepsilon = \frac{a}{b}$ with a, b coprime and $b \geq N$. In this case, two frequencies are aliased modulo p, so that $\omega_2 = \omega_1 + np$ for some nonzero $n \in \mathbb{Z}$ with $-\frac{N}{2p} \leq n < \frac{N}{2p}$. Moreover, for $h = \omega_1 \pmod{p} = \omega_2 \pmod{p}$, we have

$$\begin{aligned}
\left|\widehat{\boldsymbol{S}}_{p}[h]\right| &= |a_1 + a_2|\,, \\
\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right| &= \left|a_1 \mathrm{e}^{2\pi \mathrm{i} \omega_1 \varepsilon} + a_2 \mathrm{e}^{2\pi \mathrm{i} \omega_2 \varepsilon}\right| \\
&= \left|a_1 + a_2 \mathrm{e}^{2\pi \mathrm{i} \varepsilon (\omega_2 - \omega_1)}\right| \\
&= \left|a_1 + a_2 \mathrm{e}^{2\pi \mathrm{i} \varepsilon np}\right|. \qquad (2.16)
\end{aligned}$$

Since the coefficients a_j are assumed to be real, for the remark to hold it suffices that εnp be nonintegral. This is clear since $\varepsilon np = a\frac{np}{b}$ and by assumption $-\frac{1}{2} \leq \frac{np}{b} < \frac{1}{2}$.

Lemma 2.2.2 allows us to determine whether aliasing has occurred by checking whether $\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right| / \left|\widehat{\boldsymbol{S}}_{p}[h]\right| = 1$ for a few values of ε. It offers both a deterministic (part (A)) and a random (part (B)) procedure to identify aliasing in the sub-sampled DFTs. In practice we use a test related to part (B), and need to set a tolerance τ in order to accept or reject frequencies according to the criterion

$$\left| \frac{\left|\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]\right|}{\left|\widehat{\boldsymbol{S}}_{p}[h]\right|} - 1 \right| \leq \tau. \qquad (2.17)$$

In our algorithms we choose $\varepsilon = 1/cN$ for some small constant $c \geq 1$, which would satisfy the hypothesis of part (A) of Lemma 2.2.2. A tolerance τ on the order of p/N works well in general, which is what we use in our experiments in section 2.5 below.

In our algorithms we will take a number of sub-sampled DFTs of an input signal $S(t)$ of the form (2.1), whose lengths we denote p_ℓ. Lemma 2.2.2 allows us to determine whether or not two or more frequencies are aliased $\pmod{p_\ell}$, so that we only add non-aliased terms to our representation. Since it is unlikely that two or more frequencies are aliased modulo two different sampling rates, using a different p_ℓ in a subsequent iteration lets us quickly discover all frequencies present in $S(t)$. Lemma 2.2.6 gives a worst-case bound on the number of p_ℓ's required by our deterministic algorithm to identify all k frequencies in a given Fourier-sparse signal. It is similar to [28, Lemma 1], but with a smaller constant. In its proof we use the CRT, which we quote here for completeness (see, e.g., [38]).

Theorem 2.2.5 (Chinese Remainder Theorem)**.** *Any integer n is uniquely specified modulo N by its remainders modulo m pairwise relatively prime numbers p_ℓ, provided $\prod_{\ell=1}^{m} p_\ell \geq N$.*

Lemma 2.2.6. *Let $M > 1$. It suffices to take $1 + (k-1)\lfloor \log_M N \rfloor$ pairwise relatively prime p_ℓ's with $p_\ell \geq M$ to ensure that each frequency ω_j is isolated (i.e. not aliased) $\pmod{p_\ell}$ for at least one ℓ.*

Proof. Assume otherwise, namely that given p_ℓ for $\ell = 1, 2, \ldots, L$ with $L > k\lfloor \log_M N \rfloor$ there exists some ω_j such that ω_j is aliased $\pmod{p_\ell}$. By the Pigeon Hole Principle there exists at least one $\omega_m \neq \omega_j$ such that $\omega_j - \omega_m \equiv 0 \pmod{p_\ell}$ at least q times, where $q > \lfloor \log_M N \rfloor$. Without loss of generality we assume that $\omega_j - \omega_m \equiv 0 \pmod{p_\ell}$ for $\ell = 1, 2, \ldots, q$. Now by the fact that $p_\ell \geq M$ we have

$$\prod_{\ell=1}^{q} p_\ell \geq M^q \geq N.$$

By the CRT we would then have $\omega_j \equiv \omega_m \pmod{N}$, a contradiction. □

Remark 2.2.7. The algorithm in [28] requires taking $1+2k\log_k N$ co-prime sample lengths, since that algorithm requires each ω to be isolated in at least half of the DFTs of length p_ℓ. This requirement stems from the fact that that algorithm cannot distinguish between aliased and non-aliased frequencies in a given sub-sampled DFT. Our worst-case bound is approximately a factor of two better, though in practice our algorithms never use all those sample lengths on random input. The fact that we can tell which frequencies are "good" for a given p_ℓ allows us to construct our Fourier representation one term at a time, and quit when we have achieved a prescribed stopping criterion.

2.3 Algorithms

Both of our algorithms proceed along a similar course; in fact they differ only in the choice of the sample lengths p_ℓ. We assume that we are given access to the continuous-time signal $S(t)$ whose Fourier coefficients we would like to determine, and further that we can sample from S at arbitrary points t in unit time. This is an appropriate model for analog signals, but not for discrete ones. In the discrete case, one could interpolate between given samples to approximate the required S-values, though we have not implemented or analyzed this case. (The same assumptions hold for the algorithms in [28], while those in [20, 21, 26] are formulated purely in the discrete realm.) In this chapter we limit ourselves to the noiseless case. Though this is a highly unrealistic assumption, it permits a simple description of the underlying algorithm. In chapter we address the issue of noise specifically and give a number of modifications to our algorithms to handle noise levels of varying strengths.

2.3.1 Non-adaptive

Our algorithms start by choosing a sample length p_1 such that $p_1 \geq ck$ for some constant $c > 1$. For a fixed $\varepsilon \leq 1/N$, we then compute $\widehat{\boldsymbol{S}}_p$ and $\widehat{\boldsymbol{S}}_{p,\varepsilon}$, sort the results by magnitude, and compute frequencies ω via (2.11) for the k largest coefficients in absolute value. We then check whether or not each of those frequencies is aliased via (2.17), and if it is not, we add it to our list. The coefficient is given by the unshifted sample value $\widehat{\boldsymbol{S}}_p[h]$ at that frequency. After this, we combine terms with the same frequency and prune small coefficients from the list. We then iterate until a stopping criterion is reached. In the empirical study described in section 2.5, we stopped when the number of distinct frequencies in our list equaled the desired sparsity.

Our deterministic algorithm chooses p_ℓ to be the ℓ^{th} prime greater than ck. This ensures that all samples lengths are co-prime, at the expense of taking slightly more samples than necessary. By Lemma 2.2.6, $1 + (k-1)\lfloor \log_{ck} N \rfloor$ such p_ℓs suffice to isolate every ω at least once. This gives us worst-case sampling and runtime complexity on the same order as [28], though the results in section 2.5 indicate that on average we significantly outperform those pessimistic bounds.

Our Las Vegas algorithm chooses p_ℓ uniformly at random from the interval $[c_1 k, c_2 k]$ for constants $1 < c_1 < c_2$. In this case we cannot make a worst-case guarantee on the number of iterations needed by the algorithm to converge. However, the results in section 2.5 indicate that the Las Vegas version performs similarly to the deterministic version on the class of signals tested.

2.3.2 Adaptive

The algorithms can also be implemented in an adaptive fashion, by which we mean that the size of the current representation is taken into account in subsequent iterations. In particular, if $\boldsymbol{R}$ is our current representation, we let $k^* = k - |\boldsymbol{R}|$ and choose the next p_ℓ with respect to k^* instead of k. Moreover, before taking DFTs, we subtract off the contribution from the current representation, so that effort is not expended re-identifying portions of the spectrum already discovered. This idea is similar to that in [20, 21], though in our empirical studies the evaluation of the representation is done directly, rather than as an unequally-spaced FFT. This gives our algorithms asymptotically slower runtime, but the effect is negligible for the values of k studied in section 2.5. A formal description appears below in algorithm 1.

2.4 Average-case analysis

In this section we prove that the average-case runtime and sampling complexity of our algorithm are $\Theta(k \log k)$ and $\Theta(k)$, respectively. This is shown over a class of random signals described in section 2.4.2. Before giving this result on the expected runtime and sampling complexity, in section 2.4.1 we estimate the cost of a single iteration of the while loop in algorithm 1, lines 5–25. We then describe in section 2.4.2 the random signal model over which we prove our average-case bounds. In section 2.4.3 we prove that the expected number of iterations of the while loop is constant, and in section 2.4.4 we use this result to prove our average-case bounds.

Algorithm 1 PHASESHIFT

Input: function pointer S, integers c_1, c_2, k, N, real ε
Output: $\boldsymbol{R}$, a sparse representation for $\widehat{S}$
$\boldsymbol{R} \leftarrow \emptyset$, $\varepsilon_0 \leftarrow 0, \varepsilon_1 \leftarrow \varepsilon$, $\ell \leftarrow 1$
while $|\boldsymbol{R}| < k$ **do**
  $k^* \leftarrow k - |\boldsymbol{R}|$ {or k if non-adaptive}
  $p_\ell \leftarrow$ first prime $\geq c_1 k^*$
    {or UNIFORM$(c_1 k^*, c_2 k^*)$ if Las Vegas}
  for $m = 0$ to 1 **do**
    for $j = 0$ to $p_\ell - 1$ **do**
      $\boldsymbol{S}_{\ell,m}[j] \leftarrow S\left(\frac{j}{p_\ell} + \varepsilon_m\right)$
      $\boldsymbol{S}_{\text{rep}}[j] \leftarrow \sum_{(\omega, c_\omega) \in \boldsymbol{R}} c_\omega \mathrm{e}^{2\pi \mathrm{i} \omega (j/p_\ell + \varepsilon_m)}$
        {omit if non-adaptive}
    end for
    $\widehat{\boldsymbol{S}}_{\ell,m} \leftarrow \mathrm{FFT}(\boldsymbol{S}_{\ell,m} - \boldsymbol{S}_{\text{rep}})$
    $\widehat{\boldsymbol{S}}^{\text{sort}}_{\ell,m} \leftarrow$ SORT$(\widehat{\boldsymbol{S}}_{\ell,m})$
    for $j = 1$ to k^* **do**
      $\omega_{j,\ell} \leftarrow \frac{1}{2\pi\varepsilon} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}^{\text{sort}}_{\ell,1}[j]}{\widehat{\boldsymbol{S}}^{\text{sort}}_{\ell,0}[j]}\right)$
    end for
  end for
  for $j = 1$ to k^* **do**
    if $\left|\frac{\left|\widehat{\boldsymbol{S}}^{\text{sort}}_{\ell,0}[j]\right|}{\left|\widehat{\boldsymbol{S}}^{\text{sort}}_{\ell,1}[j]\right|} - 1\right| < \frac{p_\ell}{N}$ **then**
      $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup \left\{\left(\omega_{j,\ell}, \widehat{\boldsymbol{S}}_{\ell,0}[\omega_{j,\ell}]\right)\right\}$
    end if
  end for
  collect terms in $\boldsymbol{R}$ with same ω
  prune small coefficients from $\boldsymbol{R}$
  $\ell \leftarrow \ell + 1$
end while

2.4.1 While loop runtime and sampling complexity

The computational cost of the while loop in algorithm 1, lines 5–25 is dominated by three operations. The first is the evaluation of the current representation $\boldsymbol{R}$ of $k - k^*$ terms at the $\mathrm{O}(k^*)$ points j/p_ℓ in line 10. In our implementation, we simply calculated this directly, looping over both the sample points and the terms in the representation. The complexity of this implementation is $\mathrm{O}(p_\ell(k-k^*)) = \mathrm{O}(k^*(k-k^*)) = \mathrm{O}(k^2)$, and while non-equispaced fast Fourier transforms [18, 4] yield an asymptotically faster runtime of $\mathrm{O}(k\log(k))$, they also incur large overhead costs. For the values of k considered in this dissertation, the direct evaluation seems to have little effect on the overall runtime. The other two dominant computational tasks in the inner loop are the FFTs of $\mathrm{O}(k)$ samples and the subsequent sorting of these DFT coefficients. It is well-known that both of these operations can be done in time $\Theta(k\log(k))$ [13]. Thus the inner loop has overall time complexity $\Theta(k\log(k))$, assuming the use of non-equispaced FFTs.

2.4.2 Random signal model

For both the average-case analysis and for the empirical evaluation described in section 2.5 we considered test signals with uniformly random phase over the bandwidth and coefficients chosen uniformly from the complex unit circle. In other words, given k and N, we choose k frequencies ω_j uniformly at random (without replacement) from $[-N/2, N/2) \cap \mathbb{Z}$. The corresponding Fourier coefficients a_j are of the form $\mathrm{e}^{2\pi\mathrm{i}\theta_j}$, where θ_j is drawn uniformly from $[0, 1)$. The signal is then given by

$$S(t) = \sum_{j=1}^{k} a_j \mathrm{e}^{2\pi\mathrm{i}\omega_j t}. \tag{2.18}$$

This is the standard signal model considered in previous empirical evaluations of sub-linear Fourier algorithms [30, 28, 26].

2.4.3 Markov analysis of collisions

In order to analyze the expected runtime and sampling complexity of our algorithms, we must estimate the expected number of collisions among frequencies modulo the sample lengths used by the algorithms. Recall that in the noiseless case, our algorithms are able to detect when a collision between two or more frequencies has occurred, and for those that are not aliased we are able to calculate the value of the frequency. Thus we seek to estimate the expected fraction of frequencies that are aliased modulo a given sample length p, since this determines how many passes the algorithm makes. In this section we derive bounds on the expected value of this quantity.

In the random signal model considered in section 2.4.2, we assume the k frequencies are uniformly distributed over the bandwidth $[-N/2, N/2)$, and so the residues $\omega \bmod p$ are also uniformly distributed over $[0, p-1] \cap \mathbb{Z}$. Our problem then becomes a classical occupancy problem: the number of collisions among the frequencies is equivalent to the number of multiple-occupancy bins when k balls are thrown uniformly at random into p bins. Define X_m to be the number of single-occupancy bins after m balls are thrown, Y_m to be the number of multiple-occupancy bins after m balls are thrown, and Z_m to be the number of zero-occupancy bins after m balls are thrown. Since p is constant, we have the trivial relationship $Z_m = p - X_m - Y_m$, so it suffices to consider only the pair (X_m, Y_m).

When the $(m+1)^{\text{st}}$ ball is thrown, we have the following possibilities:

- it lands in an unoccupied bucket, with probability $Z_m/p = 1 - (X_m + Y_m)/p$;

- it lands in a single-occupancy bucket, with probability X_m/p;

- it lands in a multiple-occupancy bucket, with probability Y_m/p.

In the first case, we have $X_{m+1} = X_m + 1,\ Y_{m+1} = Y_m$; in the second case, we have $X_{m+1} = X_m - 1,\ Y_{m+1} = Y_m + 1$; and in the third case, we have $X_{m+1} = X_m,\ Y_{m+1} = Y_m$. Conditioning on the values of X_m, Y_m we have

$$\mathbb{E}\left(\left[\begin{array}{c} X_{m+1} \\ Y_{m+1} \end{array}\right] \middle| \left[\begin{array}{c} X_m \\ Y_m \end{array}\right]\right) = \left[\begin{array}{cc} 1-2/p & -1/p \\ 1/p & 1 \end{array}\right] \left[\begin{array}{c} X_m \\ Y_m \end{array}\right] + \left[\begin{array}{c} 1 \\ 0 \end{array}\right], \tag{2.19}$$

so that the system forms a Markov chain. By recursively conditioning on the values of X_{m-1}, Y_{m-1}, we can calculate the expected values of X_k, Y_k for any $k > 0$ using the initial condition $X_1 = 1$, $Y_1 = 0$.

Denoting by A the matrix in the right-hand side of equation (2.19), we have

$$\mathbb{E}\left(\left[\begin{array}{c} X_k \\ Y_k \end{array}\right]\right) = \sum_{m=0}^{k-1} \left(A^m \left[\begin{array}{c} 1 \\ 0 \end{array}\right]\right) = \left(\sum_{m=0}^{k-1} A^m\right) \left[\begin{array}{c} 1 \\ 0 \end{array}\right]. \tag{2.20}$$

Since $\rho(A) = 1 - 1/p < 1$, where ρ is the spectral radius, the geometric matrix series can be written

$$\sum_{m=0}^{k-1} A^m = (I - A)^{-1}(I - A^k). \tag{2.21}$$

After some linear algebra, we obtain

$$\mathbb{E}\left(\left[\begin{array}{c} X_k \\ Y_k \end{array}\right]\right) = \left[\begin{array}{c} k(1-\frac{1}{p})^{k-1} \\ p(1-(1-\frac{1}{p})^k) - k(1-\frac{1}{p})^{k-1} \end{array}\right]. \tag{2.22}$$

Since $Z_k = p - X_k - Y_k$, we have $\mathbb{E}\left(Z_k\right) = p(1-1/p)^k$.

In our algorithms we choose $p = ck$ for some small integer c. Using this and the approximation $(1+\frac{x}{n})^n \approx \mathrm{e}^x$, we have

$$\mathbb{E}\left(\begin{bmatrix} X_k \\ Y_k \end{bmatrix}\right) \approx \begin{bmatrix} k\mathrm{e}^{-1/c} \\ ck(1-\mathrm{e}^{-1/c}) - k\mathrm{e}^{-1/c} \end{bmatrix}. \tag{2.23}$$

This gives a nonlinear equation for the expected number of collisions among k frequencies as a function of the parameter c. Newton's method can then be used to determine the value c required to ensure a desired fraction of the frequencies are not aliased. For example, to ensure that 90% of frequencies are isolated on average, it suffices to take $c = 5$; this value for the parameter c had already been found to give good performance in our empirical evaluation of the algorithms.

2.4.4 Average-case runtime and sampling complexity

In this section we will use a probabilistic recurrence relation due to Karp [31, 17] to give average-case performance bounds and concentration results for the case when the algorithm is halted after identifying k or more terms. In particular, we use the following theorem for recurrences of the form

$$T(k) = a(k) + T(H(k)), \tag{2.24}$$

where $T(k)$ denotes the time required to solve an instance of size k, $a(k)$ is the amount of work done on a problem of size k, and $0 \leq H(k) \leq k$ is a random variable denoting the size of the subproblem generated by the algorithm.

Theorem 2.4.1. *[31, Theorem 1.2] Suppose $a(k)$ is nondecreasing, continuous, and strictly increasing on $\{x : a(x) > 0\}$, and that $\mathbb{E}[H(k)] \leq m(k)$ for a nondecreasing continuous function $m(k)$ such that $m(k)/k$ is also nondecreasing. Denote by $u(k)$ the solution to the deterministic recurrence*

$$u(k) = a(k) + u(m(k)). \tag{2.25}$$

Then for $k > 0$ and $t \in \mathbb{N}$,

$$\mathbb{P}[T(k) > u(k) + ta(k)] \leq \left(\frac{m(k)}{k}\right)^t. \tag{2.26}$$

In section 2.4.1 we showed that our algorithm does work $a(k) = \Theta(k \log(k))$ on input of size k, while in section 2.4.3 we showed that it generates a subproblem whose average size is $m(k) = k/10$. (Recall that with the parameter $c = 5$, on average over 90% of the frequencies were not aliased modulo $p = \mathrm{O}(ck)$.) The associated deterministic recurrence is then

$$u(k) = \Theta(k \log(k)) + u(k/10), \tag{2.27}$$

whose solution is $u(k) = \Theta(k \log(k))$ (see, e.g., [13]). A straightforward application of Theorem 2.4.1 yields

$$\mathbb{P}[T(k) > \Theta(k \log(k)) + tk \log(k)] \leq 10^{-t}, \tag{2.28}$$

so that the runtime is tightly concentrated about its mean $\Theta(k \log(k))$.

The sampling complexity $S(k)$ can be handled in an analogous manner, since in this case

$a(k) = \Theta(k)$ and $m(k) = k/10$ as before. The associated deterministic recurrence becomes

$$u(k) = \Theta(k) + u(k/10), \tag{2.29}$$

whose solution is $u(k) = \Theta(k)$. Applying Theorem 2.4.1 again we have

$$\mathbb{P}[S(k) > \Theta(k) + tk] \leq 10^{-t}, \tag{2.30}$$

so that we again have tight concentration around the mean $\Theta(k)$.

2.5 Empirical Evaluation

In this section we describe the results of an empirical evaluation of the *adaptive* deterministic and Las Vegas variants of the Phaseshift algorithm described above. Both algorithms were implemented in C++ using FFTW 3.0 [19] for the FFTs, using `FFTW_ESTIMATE` plans since the sample lengths are not known in advance for the Las Vegas variant. For comparison we also ran the same tests on the four variants of GFFT as well as on AAFFT and FFTW itself. The FFTW runs utilized the `FFTW_PATIENT` plans with wisdom enabled, and so are highly optimized. The experiments were run on a single core of an Intel Xeon E5620 CPU with a clock speed of 2.4 GHz and 24 GB of RAM, running SUSE Linux with kernel 2.6.16.60-0.81.2-smp for x86_64. All code was compiled with the Intel compiler using the `-fast` optimization. As in [29], timing is reported in CPU ticks using the `cycle.h` file included with the source code for FFTW.

In the following sections we refer to our algorithm as "Phaseshift", since by taking shifted time samples of the input signal we also shift the phase of the Fourier coefficients. To keep

Table 2.1: Implementations used in the empirical evaluation.

Algorithm	R/D	Samples	Runtime	Reference
PS-Det	D	k	$k \log k$	Section 2.4
PS-LV	R	k	$k \log k$	Section 2.4
GFFT-DF	D	$k^2 \log^4 N$	$k^2 \log^4 N$	[29]
GFFT-DS	D	$k^2 \log^2 N$	$Nk \log^2 N$	[29]
GFFT-RF	R	$k \log^4 N$	$k \log^5 N$	[29]
GFFT-RS	R	$k \log^2 N$	$N \log N$	[29]
AAFFT	R	$k \log^c N$	$k \log^c N$	[21]
FFTW	D	N	$N \log N$	[19]

the plots readable, we only show data for the adaptive, deterministic variant of our algorithm; the other variants perform similarly. The algorithms of [29] are denoted GFFT-XY, where X $\in$ {D,R} and Y $\in$ {F,S}. The D/R stands for deterministic or randomized, while the F/S stands for fast or slow. The fast variants use more samples but less runtime while the slow variants use fewer samples but more runtime. In the plots below, we always show the GFFT variant with the most favorable sampling or runtime complexity. Finally, AAFFT denotes the algorithm of [21]. The implementations tested are summarized in table 2.1 along with the average-case sampling and runtime complexities, and the associated references.

2.5.1 Setup

Each data point in Fig. 2.1–2.2 is the average of 100 independent trials of the associated algorithm for the given values of the bandwidth N and the sparsity k. The lower and upper bars associated with each data point represent the minimum and maximum number of samples or runtime of the algorithm over the 100 test functions. The values of k tested were $2, 4, 8, \ldots, 4096$, while the values of N were $2^{17}, 2^{18}, \ldots, 2^{26}$. For the larger values of k, the slow GFFT variants and AAFFT took too long to complete on our hardware, so we only present partial data for these algorithms. Nevertheless, the trend seen in the plots

below continues for higher values of the sparsity. The test signals were generated according to the signal model described in section 2.4.2.

The Phaseshift and deterministic GFFT variants will always recover such signals exactly. The randomized GFFT variants are Monte Carlo algorithms, and so, when they succeed, will also recover the signal exactly. AAFFT, on the other hand, is an approximation algorithm which will fail on a non-negligible set of input signals. However, for the runs depicted in Fig. 2.1–2.2, AAFFT always produced an answer with ℓ_2 error less than 10^{-4}. The randomized GFFT variants failed a total of 7 times out of 2200 test signals, a relatively small amount that can be reduced by parameter tuning. For the Phaseshift variants, we chose the parameters $c_1 = 5,\ c_2 = 10$, and took the shift ε to be $1/2N$. Finally, for the randomized GFFT variants, we chose the Monte Carlo parameter to be 1.2.

2.5.2 Sampling Complexity

In Fig. 2.1 (a), we compare the average number of samples of the input signal S required by each algorithm when the bandwidth N is fixed at 2^{22}. The sparsity of the test signal was varied from 2 to 4096 by powers of two. We can see that the Phaseshift variants require over an order of magnitude fewer samples than GFFT-RS, the GFFT variant with the lowest sampling requirements. Both Phaseshift variants also require over an order of magnitude fewer samples than AAFFT. The comparison with the deterministic GFFT variants is even starker; Phaseshift-Det requires two orders of magnitude fewer samples than GFFT-DS (not shown), and four orders of magnitude fewer samples than GFFT-DF (not shown).

In Fig. 2.2 (a), we compare the average number of samples of the input signal S required by each algorithm when the sparsity k is fixed at 60. The bandwidth N was varied from 2^{17}–

2^{26} by powers of two. Using powers of two for the bandwidth allows the best performance for both FFTW and AAFFT, though this fact is more relevant for the runtime comparisons in the following section. We can see that the Phaseshift variants require many fewer samples than all four GFFT variants as well as AAFFT and FFTW, for all values of N tested. The Phaseshift variants exhibit almost no dependence on the bandwidth for all values of N, a feature not shared by the other deterministic algorithms.

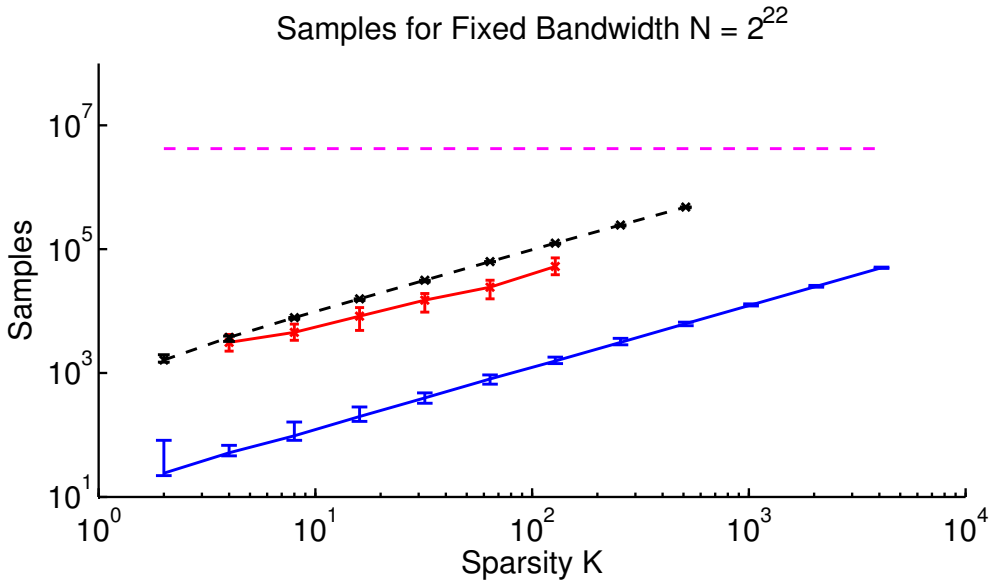
2.5.3 Runtime Complexity

In Fig. 2.1 (b), we compare the average runtime of each algorithm over 100 test signals when the bandwidth N is fixed at 2^{22}. The range of sparsity k considered is the same as in section 2.5.2. For all values of k the Phaseshift variants are faster than GFFT-RF (the fastest GFFT variant) and AAFFT by more than an order of magnitude. When compared to GFFT-RS (not shown), GFFT-DS (not shown), and FFTW, the difference in runtime is closer to three orders of magnitude.

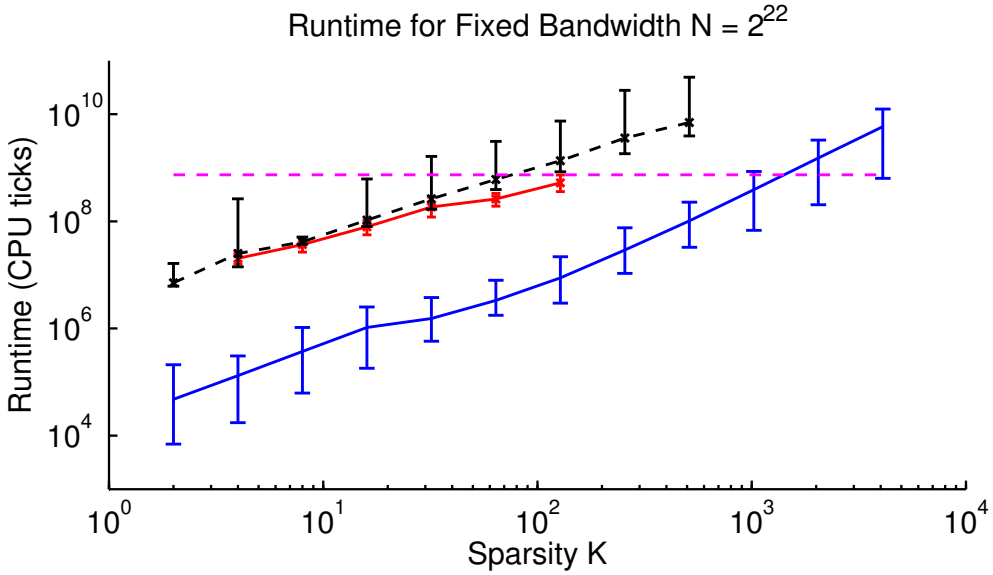
In Fig. 2.2 (b), we compare the average runtime of each algorithm over 100 test signals when the sparsity k is fixed at 60. The range of bandwidth considered is the same as in section 2.5.2. The Phaseshift variants are the only algorithms that outperform FFTW for all values of N tested. The other implementations tested only become competitive with the standard FFT for $N \gtrsim 2^{20}$, while ours are faster even for modest N.

2.6 Multiple Shifts

Our algorithm requires access to samples of the input function with a very small time shift $\varepsilon < 1/N$, where N is the assumed bandwidth of the signal. For signals with a very large

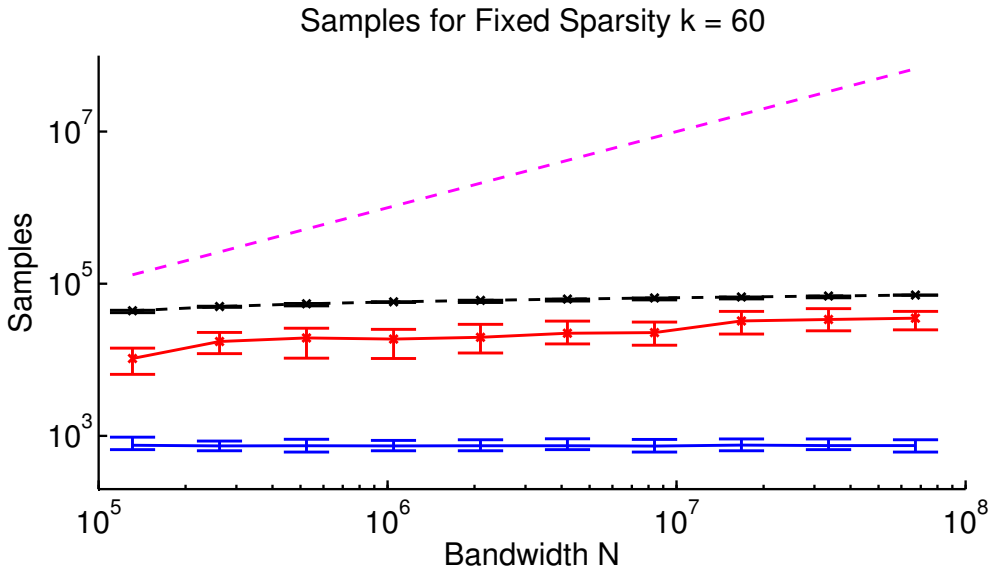


(a)

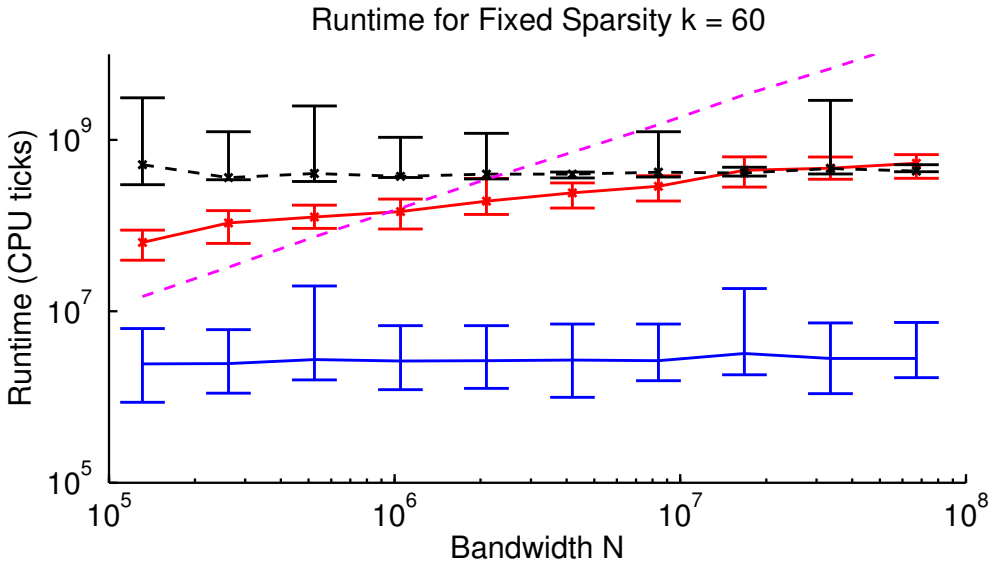


(b)

Figure 2.1: (a) Sampling complexity with fixed bandwidth $N = 2^{22}$ for PS-Det (blue solid line), GFFT-RS (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line). (b) Runtime complexity with fixed bandwidth $N = 2^{22}$ for PS-Det (blue solid line), GFFT-RF (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line). For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.



(a)



(b)

Figure 2.2: (a) Sampling complexity with fixed sparsity $k = 60$ for PS-Det (blue solid line), GFFT-RS (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line). (b) Runtime complexity with fixed sparsity $k = 60$ for PS-Det (blue solid line), GFFT-RF (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line).

bandwidth $N \gtrsim 10^{12}$, this time shift is impractical to implement with current hardware. Following [47], however, it is possible to combine two or more much smaller shifts ε_1, ε_2 to achieve the same effect.

2.6.1 Two shifts

Let r_1, r_2 be co-prime integers with $r_1 r_2 \geq N$, and set $\varepsilon_i = 1/r_i$ for $i = 1, 2$. Since $|\omega| > r_i$ in general, we no longer have the relationship

$$\omega = \frac{1}{2\pi\varepsilon_i} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{p,\varepsilon_i}[h]}{\widehat{\boldsymbol{S}}_p[h]}\right) \tag{2.31}$$

for $h = \omega \pmod{p}$ due to the wrap-around of the frequencies $\pmod{r_i}$. Rather, we consider the related quantities

$$\sigma_i \stackrel{\text{def}}{=} \frac{1}{2\pi} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{p,\varepsilon_i}[h]}{\widehat{\boldsymbol{S}}_p[h]}\right), \tag{2.32}$$

so that $\sigma_i \in [-1/2, 1/2)$. We then have

$$\sigma_i = \frac{\omega}{r_i} + k_i \tag{2.33}$$

for some $k_i \in \mathbb{Z}$. Determining ω is then equivalent to determining the values k_i.

Note that (2.33) gives two equations, one for each index i. Solving these both for ω and setting them equal gives the linear diophantine equation

$$r_1 k_1 - r_2 k_2 = r_1 \sigma_1 - r_2 \sigma_2, \tag{2.34}$$

which is solvable for (k_1, k_2) with any integer right-hand side since r_1, r_2 were taken to be

co-prime. (See, e.g., [38] for a basic treatment of linear diophantine equations.) Indeed, let b be the right-hand side of (2.34), and let (x, y) be integers such that

$$r_1 x + r_2 y = 1. \tag{2.35}$$

(These are easily furnished by the extended Euclidean algorithm [13].) The general solution to (2.34) is given by

$$\begin{aligned} k_1 &= xb - r_2 n \\ k_2 &= -yb - r_1 n, \end{aligned} \tag{2.36}$$

for any integer n. Plugging the first of these in to (2.33) and solving for ω, we have

$$\omega = r_1 \sigma_1 - r_1 bx - r_1 r_2 n. \tag{2.37}$$

Since $r_1 r_2 \geq N$ and different solutions ω differ by at least $r_1 r_2$, there is a unique n such that $-N/2 \leq \omega < N/2$. Knowing this, we can determine the value of ω mod N efficiently. Specifically, n must take one of the values

$$\left\lfloor \frac{r_1 \sigma_1 - r_1 bx}{r_1 r_2} \right\rfloor \text{ or } \left\lceil \frac{r_1 \sigma_1 - r_1 bx}{r_1 r_2} \right\rceil,$$

so we can simply check if the first of these puts ω in the correct range, and if not take the second.

The only requirements on r_1, r_2 are that they be co-prime with product greater than N. In particular, we can take $r_1 = \left\lceil N^{1/2} \right\rceil$ and $r_2 = r_1 + 1$, so that the shifts required by the

algorithm are of the order $N^{-1/2}$ instead of N^{-1} when using only one ε. For $N = 10^{12}$, say, this allows clock speeds in the megahertz range instead of terahertz. In practice, one would prefer more separation between the values r_1 and r_2; this can easily be achieved by taking r_1 to be an even number greater than $\left\lceil N^{1/2} \right\rceil / \alpha$ and r_2 an odd prime greater than $\alpha \left\lceil N^{1/2} \right\rceil$ for some $\alpha > 1$.

2.6.2 Three or more shifts

In the previous section we saw that using two shifts ε_i reduced the clock rate requirements from $\mathrm{O}(N^{-1})$ to $\mathrm{O}(N^{-1/2})$. We will show in this section that a similar procedure with m shifts can reduce the clock rate to $\mathrm{O}(N^{-1/m})$, at the cost of solving more complicated systems than (2.34).

To this end, suppose $\{r_i\}_{i=1}^m$ are integers such that $\prod_{i=1}^m r_i \geq N$, and define $\varepsilon_i = r_i^{-1}$. Using (2.32) to define σ_i, we again have a system of equations of the form (2.33) for $i = 1, \ldots, m$. Solving these for ω and summing the resulting equations gives

$$m\omega = \sum_{i=1}^{m} r_i(\sigma_i - k_i); \tag{2.38}$$

on the other hand, multiplying any one of the equations by m and solving for ω yields

$$m\omega = mr_i(\sigma_i - k_i). \tag{2.39}$$

Choosing $i = 1$ in (2.39) and combining it with (2.38) then gives the linear diophantine

equation in m variables $(k_1, \ldots, k_m)$

$$(m-1)r_1k_1 - \sum_{i=2}^{m} r_ik_i = (m-1)r_1\sigma_1 - \sum_{i=2}^{m} r_i\sigma_i. \tag{2.40}$$

This is solvable for any integer right-hand side when $\gcd((m-1)r_1, r_2, \ldots, r_m) = 1$.

The analogous system (2.34) with two shifts admitted an efficient method to find the unique solution (k_1, k_2) such that $\omega \in [-N/2, N/2)$. In the case of three or more shifts the problem is considerably more difficult, since there are $m - 1 \geq 2$ degrees of freedom in the solution space. Moreover, distinct solutions do not give rise to values of ω separated by N or more. In fact, for any value of m one can express the solutions to (2.40) in the form

$$\boldsymbol{k} = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{d}, \tag{2.41}$$

where $\boldsymbol{k} \in \mathbb{Z}^m$ and $\boldsymbol{z} \in \mathbb{Z}^{m-1}$ are variable and $\boldsymbol{A} \in \mathbb{Q}^{m \times m-1}$ and $\boldsymbol{d} \in \mathbb{Z}^{m-1}$ are fixed [43]. In addition we have the inequality constraints

$$-\frac{N}{2r_i} + \sigma_i < k_i \leq \frac{N}{2r_i} + \sigma_i. \tag{2.42}$$

Let u_i, l_i denote respectively the upper and lower bounds on k_i from (2.42). Given these and the values d_i one can consider (2.41) as a feasible integer programming problem: does there exist $\boldsymbol{z} \in \mathbb{Z}^{m-1}$ such that

$$\begin{pmatrix} \boldsymbol{A} \\ -\boldsymbol{A} \end{pmatrix} \boldsymbol{z} \leq \begin{pmatrix} \boldsymbol{u} - \boldsymbol{d} \\ \boldsymbol{d} - \boldsymbol{l} \end{pmatrix}? \tag{2.43}$$

For fixed values of m this problem is solvable in polynomial time [34], while the general case is known to be $\mathcal{NP}$-complete [43].

2.7 Conclusion

In this chapter we presented a deterministic and Las Vegas algorithm for the sparse Fourier transform problem that empirically outperform existing algorithms in average-case sampling and runtime complexity. While our worst-case bounds do not improve the asymptotic complexity, we proved average-case bounds over a representative class of random signals that significantly improve the computational complexity of our algorithm with respect to its competitors. We are moreover able to extend by an order of magnitude the range of sparsity for which our algorithm is faster than FFTW in the average case.

The improved performance of our algorithm can be attributed to two major factors: adaptivity and ability to detect aliasing. In particular, we are able to extract more information from a small number of function samples by considering the *phase* of the DFT coefficients in addition to their magnitudes. This represents a significant improvement over the current state of the art for the sparse Fourier transform problem. Finally, we showed that our algorithm can be implemented efficiently using two large co-prime shifts in place of one very small shift. This has significant implications in practical situations where hardware with high clock rates costs significantly more than lower rate hardware.

Chapter 3

Multi-scale sub-linear Fourier algorithms for noisy data

3.1 Introduction

In chapter we developed algorithms for approximating the discrete Fourier transform of noiseless signals using much fewer computational resources than a traditional FFT. Unfortunately any signal recorded from real-world data will contain errors of one sort or another, due to imprecise instrumentation or noisy environments. In this chapter we develop modified versions of our noiseless algorithm to address the challenge of noisy, real-world data.

The remainder of this chapter is organized as follows. In section 3.2 we describe our noise model, discuss some of the problems associated with noisy signals, and argue that in certain applications the ℓ_2 error metric is inappropriate and should be replaced with a form of Earth Mover Distance. In section 3.3 we give a simple modification of our algorithm suitable for low-level noise. The runtime complexity for this algorithm is still $\Theta(k \log k)$, but with a much

larger constant than in the noiseless case. We illustrate its performance with an empirical evaluation. In section 3.4 we give another modification of our algorithm which uses multiple shifts ε_q to determine energetic frequency values in a multi-scale manner. The resulting algorithm has runtime complexity $\Theta\left(k^2 \log(k)\right)$, but is robust to higher-level noise. This is confirmed in an empirical evaluation of the two versions of our multi-scale algorithm.

3.2 Mathematical background

In this section we describe the mathematical setting for our study of sub-linear Fourier algorithms in the presence of noise. In section 3.2.1 we describe the specific noise model used in our analysis and our empirical studies, and we discuss its advantages and limitations. In section 3.2.2 we derive some fundamental results on the stability of the operations underpinning our algorithm to noise.

3.2.1 Noise model

In order to derive some analytic estimates for use in our algorithms below, we limit ourselves in this dissertation to noisy signals of a specific type. Namely, we consider i.i.d. complex gaussian noise with covariance $\Sigma = \sigma^2 I$, where I is the two-by-two identity matrix. If $S(t)$ is a signal of the form (2.1), our discrete measurements are of the form

$$\boldsymbol{S}_p^{\mathrm{n}}[j] = S\left(\frac{j}{p}\right) + \sigma\left(\eta_j^1 + \mathrm{i}\eta_j^2\right), \tag{3.1}$$

where η_j^i are i.i.d. standard normal random variables (mean zero, unit variance) for $i = 1, 2$, $j = 1, \ldots, p$ and the superscript "n" denotes that these are noisy measurements.

Noise of the form (3.1) has the advantage that it is amenable to analytic estimates, at the expense of perhaps being overly pessimistic in modeling real-world phenomena. To emphasize this last point, we note that since gaussian random variables are unbounded, there is always a small probability that the noise overwhelms the signal in a given measurement. This is not realistic for real-world signals, and in our empirical studies in sections 3.3.2 and 3.4.3 we truncate the noise to two standard deviations. We note further that uncorrelated white noise is only a model for real-world noise; depending on the application area, some dependence or color may be present.

3.2.2 Sensitivity to noise

As in section 2.2.1, we apply the DFT to our samples to approximate the significant Fourier coefficients of the underlying signal $S(t)$. With noisy measurements of the form (3.1), we have

$$\begin{aligned}
\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h] &= \sum_{j=0}^{p-1} \boldsymbol{S}_p^{\mathrm{n}}[j] \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \sum_{j=0}^{p-1} \left(S\left(\frac{j}{p}\right) + \sigma\left(\eta_j^1 + \mathrm{i}\eta_j^2\right)\right) \mathrm{e}^{-2\pi \mathrm{i} hj/p} \\
&= \widehat{\boldsymbol{S}}_p[h] + \sigma\left(\widehat{\eta}_h^1 + \mathrm{i}\widehat{\eta}_h^2\right). \qquad (3.2)
\end{aligned}$$

Here we've written

$$\widehat{\eta}_h^i \stackrel{\mathrm{def}}{=} \sum_{j=0}^{p-1} \eta_j^i \mathrm{e}^{-2\pi \mathrm{i} hj/p} \qquad (3.3)$$

to denote the DFT of the noise terms η_j^i. Note that $\mathbb{E}\left[\widehat{\eta}_h^i\right] = 0$ and $\mathbb{E}\left[\left|\widehat{\eta}_h^i\right|^2\right] = p$ for $i = 1, 2$, $h = 0, \ldots, p-1$.

Since these noise terms have mean zero, we have

$$\mathbb{E}\left[\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]\right] = \widehat{\boldsymbol{S}}_p[h]. \tag{3.4}$$

Moreover, the variance of $\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]$ can be calculated as

$$\begin{aligned}
\mathrm{Var}\left[\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]\right] &= \mathbb{E}\left[\left|\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h] - \widehat{\boldsymbol{S}}_p[h]\right|^2\right] \\
&= \mathbb{E}\left[\sigma^2\left(\left|\widehat{\eta}_h^1\right|^2 + \left|\widehat{\eta}_h^2\right|^2\right)\right] \\
&= 2\sigma^2 p.
\end{aligned} \tag{3.5}$$

Thus, a typical noisy DFT coefficient $\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]$ will deviate from the true value $\widehat{\boldsymbol{S}}_p[h]$ by an amount proportional to $\sigma\sqrt{p}$. Recall from 2.2.1 that the discrete Fourier coefficients are given by $\widehat{\boldsymbol{S}}_p[h] = pa_\ell$ for $\omega_\ell \equiv h \pmod{p}$. A typical noisy DFT coefficient will therefore satisfy

$$\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h] = p\left(a_\ell + \mathrm{O}\left(\frac{\sigma}{\sqrt{p}}\right)\right). \tag{3.6}$$

The locations of the peaks of $\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]$ will therefore give the correct values of $\omega_\ell \pmod{p}$ as long as $\sigma/\sqrt{p} \ll |a_\ell|$. This will be the case in any reasonable setting, since in general there can be no hope of recovering Fourier coefficients whose magnitude is less than that of the noise.

Our noiseless algorithm relies crucially on the computation of the phase angle of complex numbers to determine the frequency index associated with a significant Fourier coefficient.

The numbers whose phase angles we are interested in calculating are the ratios $\widehat{\boldsymbol{S}}_{p,\varepsilon}[h]/\widehat{\boldsymbol{S}}_p[h]$, which must be replaced in the noisy case by $\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]/\widehat{\boldsymbol{S}}^{\mathrm{n}}_p[h]$. Suppose that $\omega \equiv h \pmod{p}$ is one of the frequencies appearing in (2.1) and let a be the corresponding Fourier coefficient (so that $a = p^{-1}\widehat{\boldsymbol{S}}_p[h]$). Using the binomial approximation $(1+z)^\alpha = 1 + \alpha z + \mathrm{O}\left(z^2\right)$ and a typical value for the noisy DFT coefficients, we have

$$\begin{aligned}\frac{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}^{\mathrm{n}}_p[h]} &= \frac{\widehat{\boldsymbol{S}}_p[h]\mathrm{e}^{2\pi\mathrm{i}\omega\varepsilon} + \mathrm{O}\left(\sigma\sqrt{p}\right)}{\widehat{\boldsymbol{S}}_p[h] + \mathrm{O}\left(\sigma\sqrt{p}\right)} \\ &= \frac{\mathrm{e}^{2\pi\mathrm{i}\omega\varepsilon} + \mathrm{O}\left(\sigma/a\sqrt{p}\right)}{1 + \mathrm{O}\left(\sigma/a\sqrt{p}\right)} \\ &= \mathrm{e}^{2\pi\mathrm{i}\omega\varepsilon} + \mathrm{O}\left(\sigma/a\sqrt{p}\right). \end{aligned} \tag{3.7}$$

Thus the ratio of noisy DFT coefficients agrees with the noiseless ratio up to an error term of order $\sigma/|a|\sqrt{p}$.

Given this estimate for the ratio of noisy DFT coefficients, we can derive bounds for the error in the phase angle computed via $\mathrm{Arg}(z)$. Note that $\mathrm{Arg}(z)$ is differentiable except across the negative real axis, so assume for the moment that $\mathrm{Arg}(z)$ is sufficiently far from $\pm\pi$. Since $\mathrm{Arg}(z)$ is constant on rays from the origin and linear (with unit slope) along circles centered at the origin, we have

$$|\mathrm{Arg}\,(z+\eta) - \mathrm{Arg}(z)| \leq |\eta|, \tag{3.8}$$

as long as $-\pi + |\eta| < \mathrm{Arg}(z) < \pi - |\eta|$. Combining the bound (3.8) with the estimate (3.7), we have

$$\left|\mathrm{Arg}\left(\frac{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}^{\mathrm{n}}_p[h]}\right) - 2\pi\omega\varepsilon\right| \leq \mathrm{O}\left(\frac{\sigma}{|a|\sqrt{p}}\right); \tag{3.9}$$

equivalently, if $\widetilde{\omega} = \frac{1}{2\pi\varepsilon} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p}[h]}\right)$ is our noisy frequency estimate, we have

$$|\widetilde{\omega} - \omega| \leq \mathrm{O}\left(\frac{\sigma}{2\pi\varepsilon|a|\sqrt{p}}\right). \tag{3.10}$$

For fixed ε, $|a|$, we can see from this last inequality that the ratio $\sigma/\sqrt{p}$ is critical in determining the sensitivity of our phase calculation to noise. In the remainder of this chapter we examine how the choice p and ε affect the quality of the frequency approximation in our algorithm.

3.2.3 Earth mover distance

In the existing literature on the sparse Fourier transform, the ℓ_2 norm is most often used to assess the quality of approximation. There are many reasons for this choice, with the two most convincing perhaps being the completeness of the complex exponentials with respect to the ℓ_2 norm and Parseval's theorem. For certain applications, however, this choice of norm is inappropriate. For example, in wide-band spectral estimation and radar applications, one is interested in identifying a set of frequency intervals containing active Fourier modes. In this case, an estimate $\widetilde{\omega}$ of the true frequency ω with $|\widetilde{\omega} - \omega| \ll N$ is useful, but unless $\widetilde{\omega} = \omega$ the ℓ_2 metric will report an O(1) error.

For these reasons, we propose measuring the approximation error of sparse Fourier transform problems with the Earth Mover Distance (EMD) [40]. Originally developed in the context of content-based image retrieval, EMD measures the minimum cost that must be paid (with a user-specified cost function) to transform one distribution of points into another. EMD can be calculated efficiently as the solution of a linear program corresponding to

a certain flow minimization problem. In our situation, we consider the cost to move a set of estimated Fourier modes and coefficients $\left\{(\widetilde{\omega}_j, c_{\widetilde{\omega}_j})\right\}_{j=1}^{\widetilde{k}}$ to the true values $\left\{(\omega_i, c_{\omega_j})\right\}_{j=1}^{k}$ under the cost function

$$d_1\big((\omega, c_\omega), (\widetilde{\omega}, c_{\widetilde{\omega}}); N\big) \stackrel{\text{def}}{=} \frac{|\omega - \widetilde{\omega}|}{N} + |c_\omega - c_{\widetilde{\omega}}|. \tag{3.11}$$

This choice of cost function strikes a balance between the fidelity of the frequency estimate (as a fraction of the bandwidth) and that of the coefficient estimate. We denote the EMD using d_1 for the cost function by EMD(1) in our empirical studies in sections 3.3.2 and 3.4.3 below.

3.3 A minor modification

In this section we give a simple modification of our noiseless algorithm to improve the robustness of the frequency estimation in the presence of noise. In section 3.3.1 we describe the modification and study its performance as a function of the noise and the sampling rate p. In section 3.3.2 we report the results of an empirical evaluation of the accuracy of the modified algorithm as a function of the noise level.

3.3.1 Rounding

As noted in section 3.2.2, the frequency reconstruction from noisy measurements is correct up to an error term of size $\mathrm{O}\left(\sigma/2\pi\varepsilon|a|\sqrt{p}\right)$. Moreover, the location of the peaks in the noisy DFT are robust to noise, so that the values of $\omega_\ell \pmod p$ given by the indices of the k largest coefficients in $\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[h]$ can be taken as exact. By combining these two measures we can

more reliably estimate the frequency indices of significant coefficients.

Our proposed modification is therefore to simply round the noisy frequency estimate $\widetilde{\omega} = \frac{1}{2\pi\varepsilon} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p}[h]}\right)$ to the nearest integer of the form $np + h$. This improved estimate is therefore given by

$$\widetilde{\omega}' = p \cdot \operatorname{round}\left(\frac{\widetilde{\omega} - h}{p}\right) + h, \tag{3.12}$$

where $\operatorname{round}(x)$ returns the nearest integer to x. For low levels of noise this modification will return the true value ω, while for larger noise levels it is possible that $\widetilde{\omega}$ deviates by more than $p/2$ from the true frequency ω. In this case the estimate $\widetilde{\omega}'$ will be wrong by a multiple of p. Choosing larger values for p (i.e. increasing the parameter c_1) will help to reduce the likelihood of an error in frequency estimation. Furthermore, to assure that the estimated frequencies are sufficiently far from the branch cut of $\operatorname{Arg}(\cdot)$ along the negative real axis, we take the shift $\varepsilon \leq 1/2N$. The estimated frequencies then satisfiy $-N \leq \widetilde{\omega} < N$, so that the deviations due to the noise level will not push the estimates across the discontinuity.

We saw in the previous section that the error in the phase estimation is proportional to $\sigma p^{-1/2}$, so we would expect similar behavior for the improved estimate $\widetilde{\omega}'$. To test this, we generated 100 frequencies uniformly at random from $[-N/2, N/2)$ with $N = 2^{22}$ for a range of parameters (σ, p), and set the corresponding coefficient to unity. Thus our test signals for this empirical trial were one-term trigonometric polynomials. We then reconstructed the frequencies in two ways: first, simply using the formula 2.11, and second by combining this estimate with the rounding procedure (3.12). In figure 3.1 we plot the average phase error in logarithmic scale as a function of both σ and p, which were varied from 0.001 to 0.512 and from 10 to 5120, respectively, by powers of two.

In the plot on the left, which corresponds to reconstruction using only (2.11), we can

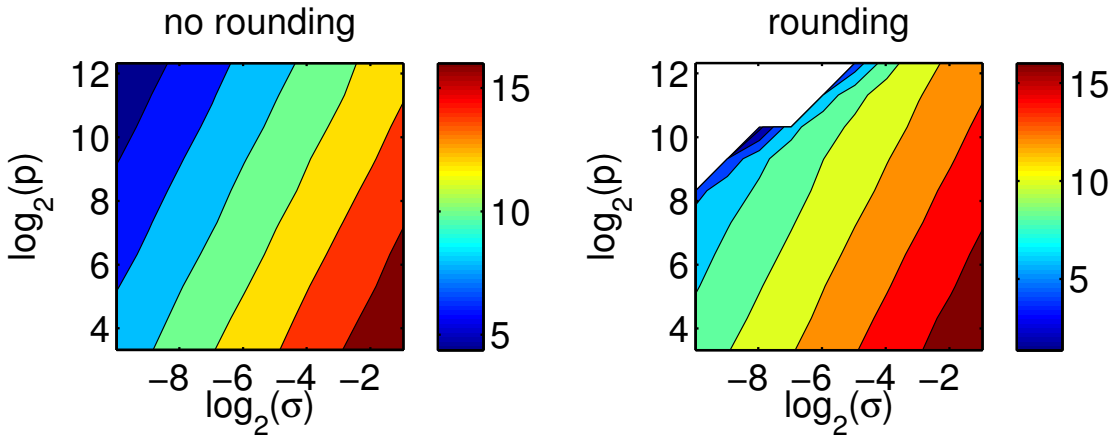


Figure 3.1: (left): Mean phase error without rounding (shaded attribute) vs. noise level σ and sample length p, in logarithmic scale. (right) Mean phase error with rounding (shaded attribute) vs. noise level σ and sample length p, in logarithmic scale. The bandwidth for both plots is $N = 2^{22}$.

clearly see the contours of constant phase error obeying the relationship $\log_2(p) = 2\log_2(\sigma) + c$ for some constant c. This confirms our analytic estimate from section 3.2.2 that the phase error is proportional to $\sigma/\sqrt{p}$. In the plot on the right, which corresponds to the improved reconstruction using (3.12), we can see that for large values of σ and small values of p the same relationship holds. However, for smaller σ and larger p we see an abrupt transition to exact reconstruction (the white area in the upper-left). The boundary of this region appears to follow the relationship $\log_2(p) = \frac{2}{3}\log_2(\sigma) + c$, indicating that for small enough values of the ratio $\sigma/p^{3/2}$ the rounding procedure is exact.

3.3.2 Empirical evaluation

In this section we describe the results of an empirical evaluation of the algorithm with the rounding procedure (3.12). This modification doesn't change the runtime or sampling complexity significantly, so we focus here on the error in the approximation as a function of the noise level σ and the parameter c_1.

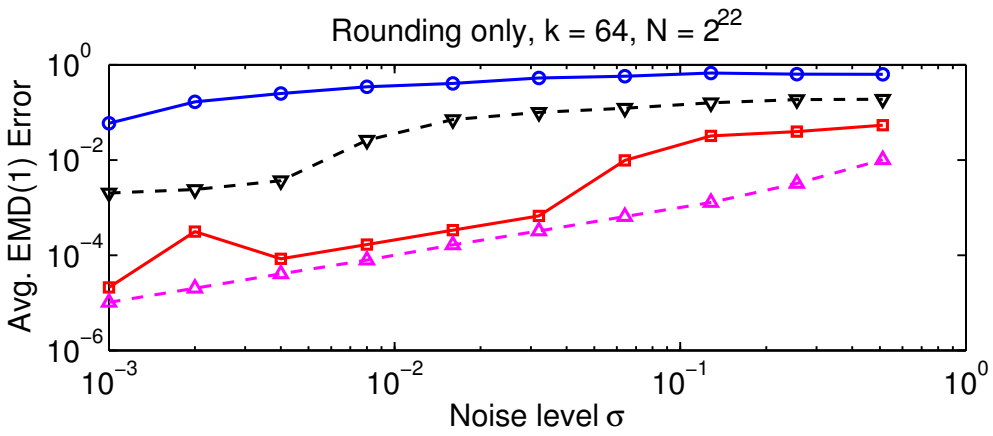


Figure 3.2: EMD(1) error as a function of the noise level σ with the sparsity and bandwidth fixed at $k = 64$, $N = 2^{22}$, respectively. Different values of the parameter c_1 are shown: $c_1 = 4$ (blue solid line), $c_1 = 16$ (black dashed line), $c_1 = 64$ (red solid line), and $c_1 = 256$ (magenta dashed line).

In figure 3.2 we report the average EMD(1) error over 100 test signals as a function of the input noise level σ (note that σ is *not* given in dB), for various choices of the parameter c_1. The test signals were generated in the same manner as in section 2.4.2: k-term trigonometric polynomials with frequencies drawn uniformly at random from $[-N/2, N/2)$ and coefficients drawn uniformly from the complex unit circle, with complex gaussian noise of variance σ added to each measurement. In this experiment, the sparsity and bandwidth are fixed at $k = 64$ and $N = 2^{22}$, respectively. As expected, the error decreases as c_1 increases, since the rounding procedure is more likely to result in the true frequency. Moreover, for larger values of c_1, the error increases linearly with the noise level, indicating the procedure's robustness in the presence of noise. For smaller values of c_1 the EMD(1) error is saturated by the frequency error, so that there is only a mild dependence on σ.

We remark that in the noiseless case the choice $c_1 = 5$ was found to be sufficient, while figure 3.2 indicates that the much larger value $c_1 \approx 256$ is necessary for good approximation in the EMD(1) metric. The larger sample lengths imply an increase in both the runtime and

sampling complexity, and indicate that the rounding procedure should be complemented by other modifications. In section 3.4 we combine the rounding procedure with the use of larger shifts ε_q in a multiscale approach to frequency estimation that is robust to higher noise levels even at coarse sampling rates (i.e. small values of c_1).

3.4 Multi-scale methods

In this section we describe two approaches to multi-scale frequency estimation using multiple shifts ε_q. These algorithms exhibit a form of error correction in their estimation of significant frequency values, which allows them to use much coarser sampling rates than the simple rounding method described in section 3.3 to achieve a desired error. This comes, however, at the cost of increased average-case runtime and sampling complexities of $\Theta\left(k^2 \log(k) \log(N)\right)$ and $\Theta\left(k^2 \log(N)\right)$, respectively. We note that the algorithms presented below, while simple to describe and implement, are not optimal in terms of computational complexity – the authors of [25] have recently given a $\Theta\left(k \log(N) \log(N/k)\right)$ time algorithm for the general (noisy) case. That algorithm, however, is considerably more complicated than ours, and moreover has not yet been shown to empirically outperform standard FFT implementations. In section 3.4.1 we give some background on our multi-scale methods and introduce the main idea of our algorithms. In section 3.4.2 we describe two variations on the basic multi-scale algorithm, and in section 3.4.3 we report the results of an empirical evaluation of the algorithms.

3.4.1 Preliminaries

In both the noiseless algorithm and the modified version given by rounding the frequencies to the nearest integers with the correct remainder modulo p, we required that $\varepsilon \leq 1/N$. This was due to the fact that the range of the principle argument function $\mathrm{Arg}(z)$ is $[-\pi, \pi)$, so to identify a frequency bandlimited to $[-N/2, N/2)$ without any wrap-around aliasing we needed to ensure that $2\pi\varepsilon\omega \in [-\pi, \pi)$. Moreover, we noted in remark 2.2.1 that, given an estimate of the magnitude of a particular frequency $|\omega| < \frac{L}{2}$, then taking $\varepsilon < \frac{1}{L}$ suffices to recover ω without wrap-around aliasing. This is the key observation for our multi-scale algorithms, which build up an estimate for ω in an iterative fashion. By increasing the shift ε while simultaneously demodulating by our current frequency estimate, we obtain a new estimate on a finer scale that allows us to correct for errors introduced in previous iterations.

3.4.2 Algorithms

Let us describe the general architecture of these algorithms in some detail. We begin by choosing an appropriate p and $\varepsilon < 1/2N$ and calculating our initial frequency estimates $\widetilde{\omega}_j^0$, $1 \leq j \leq k^*$, as in section 3.3. Thus for $\omega_j \equiv h \pmod{p}$ we have

$$\widetilde{\omega}_j^0 = p \cdot \mathrm{round}\left(\frac{1}{2\pi\varepsilon p}\left(\mathrm{Arg}\left(\frac{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p,\varepsilon}[h]}{\widehat{\boldsymbol{S}}^{\mathrm{n}}_{p}[h]}\right) - h\right)\right) + h, \tag{3.13}$$

which will in general differ from the true frequency ω_j by approximately $\sigma/2\pi\varepsilon\sqrt{p}$. For an appropriate choice of p, the difference between the estimated and true frequencies will satisfy

$$\left|\widetilde{\omega}_j^0 - \omega_j\right| \ll \left|\omega_j\right|. \tag{3.14}$$

We then begin an iterative frequency estimation procedure that tries to correct for the error in phase. For each of the k^* estimated frequencies, we increase the shift ε (say by a multiplicative factor of 2^B, where B is a parameter) and demodulate the input by our first estimate $\widetilde{\omega}_j^0$. That is, for $1 \leq j \leq k^*$ we collect the samples

$$\boldsymbol{S}_p^{\mathrm{n}}[q] = \mathrm{e}^{-2\pi \mathrm{i} \widetilde{\omega}_j^0 q/p} S\left(\frac{q}{p}\right), \tag{3.15}$$

$$\boldsymbol{S}_{p,\varepsilon}^{\mathrm{n}}[q] = \mathrm{e}^{-2\pi \mathrm{i} \widetilde{\omega}_j^0 \left(q/p + \varepsilon 2^B\right)} S\left(\frac{q}{p} + \varepsilon 2^B\right), \tag{3.16}$$

and perform DFTs on them. Note that by elementary properties of Fourier series, the peak location $h = \omega_j \pmod p$ will be shifted down by $\widetilde{\omega}_j^0 \pmod p$. From (3.13) we see that $\widetilde{\omega}_j^0 \pmod p = h$, so that the peak corresponding to ω_j will appear at the zero frequency in the demodulated signal.

Moreover, the phase difference between the shifted and unshifted DFT values at the zero frequency will be proportional to $\widetilde{\omega}_j^0 - \omega_j$, and we can estimate this phase error using the principle argument function as before. We therefore have the improved estimate

$$\widetilde{\omega}_j^1 = \widetilde{\omega}_j^0 + p \cdot \mathrm{round}\left(\frac{1}{2\pi\varepsilon 2^B p} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{p,\varepsilon}^{\mathrm{n}}[0]}{\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[0]}\right)\right). \tag{3.17}$$

Note that increasing the shift by a factor 2^B ensures that the correction term will lie in the range $\left[\frac{-1}{\varepsilon 2^B}, \frac{1}{\varepsilon 2^B}\right)$, and will differ from the true error $\widetilde{\omega}_j^0 - \omega_j$ by approximately $\sigma 2^{-B}/2\pi\varepsilon\sqrt{p}$. Thus the correction term operates on scales a factor 2^B smaller than the original estimate.

The algorithms then iterate this procedure: demodulate the input samples by the previous

frequency estimate $\widetilde{\omega}_j^{i-1}$, increase the shift by a factor 2^B, and estimate the correction term as in (3.17). In this way the estimates "zoom in" on the correct frequency value in a multi-scale manner. Since we must demodulate by different amounts for each of the k frequencies, one iteration of this procedure requires $\Theta\left(k^2\right)$ samples and $\Theta\left(k^2 \log(k)\right)$ time. Note that each iteration of the frequency estimation procedure provides a new estimate of a significant frequency's corresponding coefficient in $\widehat{\boldsymbol{S}}_p^{\mathrm{n}}[0]$. We use this fact to improve the robustness of our coefficient estimation by taking the median of the real and imaginary parts of the estimates.

Our two versions differ in how the the shifts are increased, which in turn affects how many iterations of the estimation procedure are performed. In the "non-adaptive shift" algorithm, B is a user-specified constant, so the number of iterations in the non-adaptive shift algorithm is $\lceil \log_2(N) \rceil / B$. The "adaptive shift" algorithm uses and uses the size of the previous correction term to estimate an appropriate value for ε_j^i, the the shift used in the i^{th} iteration on the j^{th} frequency. Specifically, we set

$$\varepsilon_j^i = \begin{cases} \frac{2^{\left\lceil \alpha + \log_2\left(\left|\widetilde{\omega}_j^{i-1}\right|\right)\right\rceil}}{N} & \text{if } \widetilde{\omega}_j^{i-1} \neq 0, \\ \frac{2^{\lceil \alpha + \log_2(p) \rceil}}{N} & \text{otherwise.} \end{cases} \tag{3.18}$$

The "safety parameter" $\alpha \geq 1$ guards against underestimating the magnitude of the true error. In this adaptive version we stop the estimation procedure when the correction term is zero in two consecutive iterations for all frequencies. While we cannot make an analytic claim on the number of iterations the adaptive shift algorithm makes, in practice it performs similarly to the non-adaptive version.

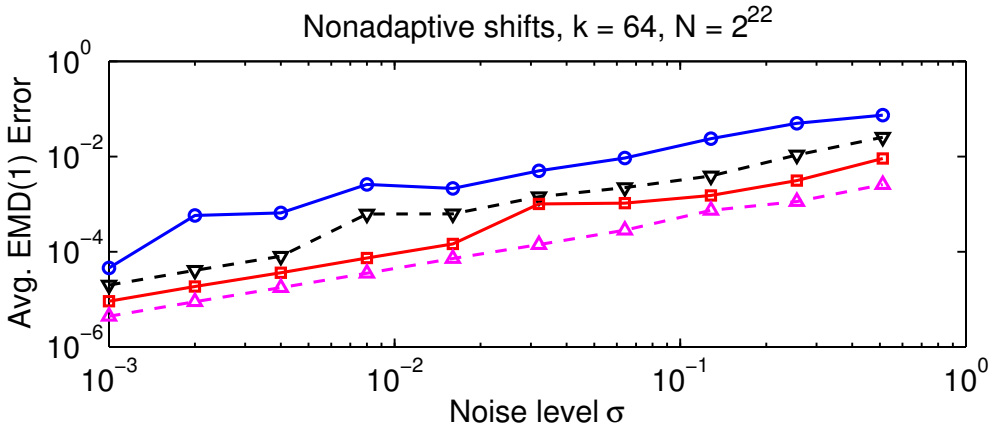
Finally, we note that for small values of the sparsity k, only requiring $p \geq c_1 k$ would

allow for very coarse sampling rates. As we saw in section 3.2.2, the frequency error is proportional to $\sigma/\varepsilon\sqrt{p}$. If p is too small, then this phase error could be very large, and it may not be the case that (3.14) holds. To guard against this we require in addition that $p \geq c\sigma^2 2^{2B}$, where the constant c is determined from a fit to the data in figure 3.1 (left).

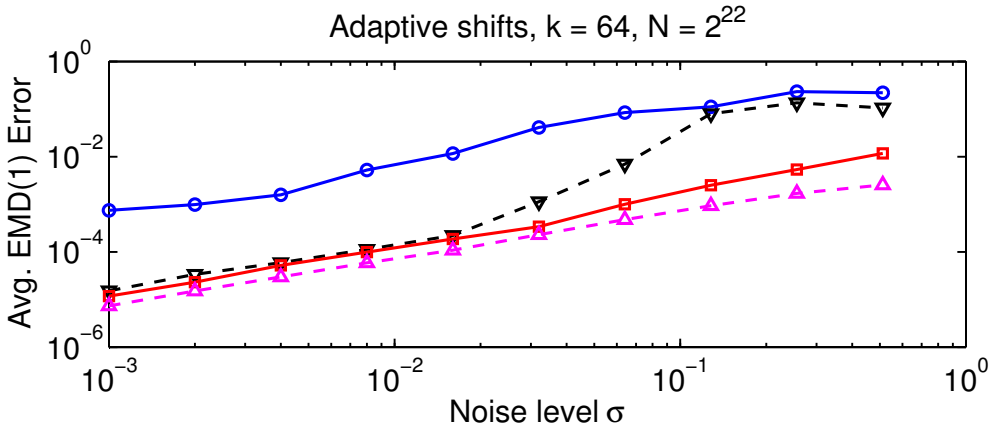
3.4.3 Empirical evaluation

To test both versions of our multi-scale algorithms, we performed an empirical evaluation with test signals identical to those in section 3.3.2. We varied the noise level and the parameter c_1 over the same range of values, set $B = 4$ in the non-adaptive shift algorithm and $\alpha = 1$ in the adaptive shift algorithm. In figure 3.3 we report the average EMD(1) error as a function of the noise level σ for different choices of the parameter c_1. The sparsity $k = 64$ and bandwidth $N = 2^{22}$ are fixed, as in figure 3.2. Note that for both the non-adaptive shift and adaptive shift algorithms the error is well-behaved for all values of c_1, whereas we had to take $c_1 \approx 256$ to achieve a similar behavior when using the rounding only algorithm of section 3.3. Thus with our multi-scale algorithms, we can achieve a prescribed error tolerance using much coarser samplings.

In figure 3.4 we report the number of samples and the runtime required for each of our three algorithms to approximate the true spectrum to within $\sigma/\sqrt{k}$ in the EMD(1) metric. In these plots, the bandwidth $N = 2^{22}$ and noise level $\sigma = 0.512$ fixed, but the parameter c_1 varies for each data point. As seen in figures 3.2 and 3.3, the rounding only algorithm generally needs a larger c_1 value to achieve the same error as the multi-scale algorithms. To make a fair comparison, we find the smallest c_1 that achieves the error bound $\sigma/\sqrt{k}$ and plot the corresponding number of samples and runtime.



(a)



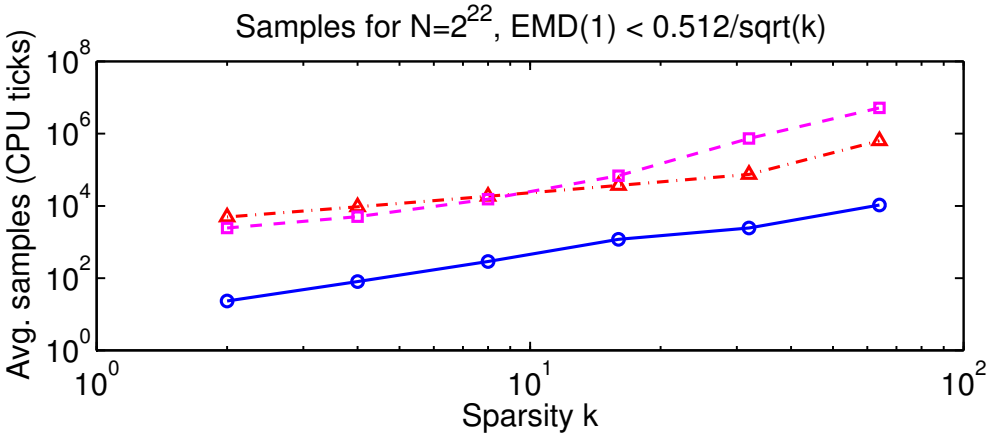
(b)

Figure 3.3: EMD(1) error as a function of the noise level σ with the sparsity and bandwidth fixed at $k = 64$, $N = 2^{22}$, respectively. Different values of the parameter c_1 are shown: $c_1 = 4$ (blue solid line), $c_1 = 16$ (black dashed line), $c_1 = 64$ (red solid line), and $c_1 = 256$ (magenta dashed line). (a) Nonadaptive shift algorithm ($B = 4$), (b) Adaptive shift algorithm ($\alpha = 1$).

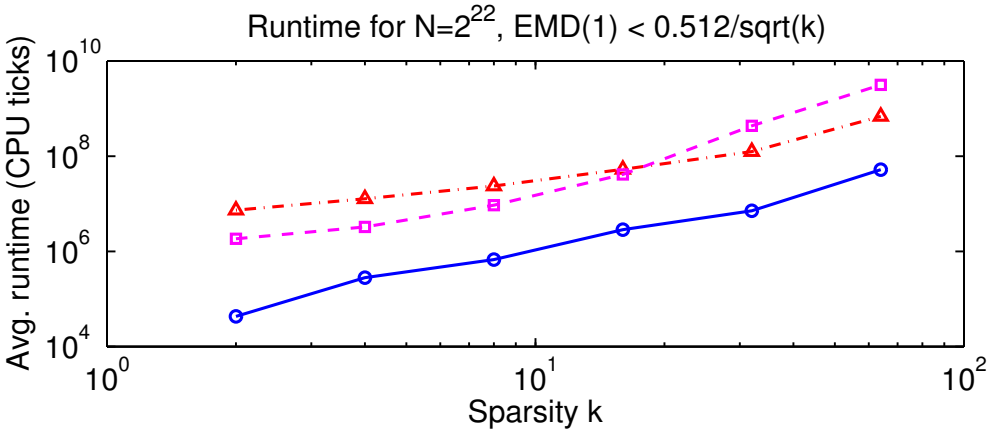
From figure 3.4 (a) we can see that the rounding only algorithm requires fewer samples to achieve the error bound than the multi-scale algorithms, even though the parameter c_1 is much larger in general. From figure 3.4 (b) we can see that the same holds true for the runtime. This is to be expected since the iterative frequency estimation procedure in the multi-scale algorithms is currently a computational bottleneck.

3.5 Conclusion

In this chapter we extended the algorithm of chapter to handle noisy signals. The first modification we gave is a minor alteration to the noiseless algorithm that doesn't change the computational complexity significantly, though it does require a relatively fine sampling rate to produce quality approximations in the EMD(1) metric. We also gave two multi-scale algorithms that exhibit error correction in their frequency estimation procedures. While these algorithms allow the use of much coarser samplings to provide good approximations, they are slower both in theory and in practice than the minor modification. We remark that our multi-scale algorithms are somewhat wasteful in their frequency estimation procedures, since currently only the values of the samples at the zero frequency are used to estimate the frequency error. It remains an open question as to whether more information can be extracted from the demodulated samples so as to reduce the complexity of the frequency estimation procedure.



(a)



(b)

Figure 3.4: (a) Samples to achieve EMD(1) error $\sigma/\sqrt{k}$, as a function of the sparsity k with fixed $\sigma = 0.512, N = 2^{22}$, for rounding only (blue solid line), non-adaptive shift (red dash-dot line), and adaptive shift (magenta dashed line) algorithms. (b) Runtime to achieve the same error bound, as a function of the sparsity k.

Chapter 4

Sub-linear Fourier algorithms in multiple dimensions

4.1 Introduction

In this chapter we shift our focus from one-dimensional signals to the multidimensional case, concentrating (as in chapter) on the noiseless case. We give three extensions of our basic one-dimensional noiseless algorithm to the multi-dimensional setting. Two of these are a result of the separability properties of the exponential function, while the third uses a number-theoretic construction to generate an equivalent one-dimensional problem on which our original algorithm can be applied. In this chapter we show that the resulting algorithms require fewer samples of the input and execute in less CPU time than the d-dimensional FFT.

The remainder of this chapter is organized as follows. In section 4.2 we define our notation for the multidimensional case. In section 4.3, we describe a straightforward extension of

our one-dimensional algorithm to multiple dimensions. The resulting algorithm is simple to describe and implement, but suffers from increased computational complexity that scales exponentially in the dimension. In order to overcome this barrier, in section 4.4 we propose an alternative method that computes the one-dimensional Fourier transform of an "unwrapped" version of the multi-dimensional signal. Finally, in section 2.5 we describe the results of an empirical evaluation of the proposed methods in two dimensions.

4.2 Notation

We denote the ambient dimension by d, and change the independent variable from t to $\boldsymbol{x}$ and use boldface to emphasize the multidimensional character of the input. We write the components of d-dimensional vectors with subscripts, so that $\boldsymbol{x} = \left(x_1, \ldots, x_d\right)^\top$. The Euclidean inner product is written

$$\boldsymbol{\omega} \cdot \boldsymbol{x} \stackrel{\text{def}}{=} \sum_{q=1}^{d} \omega_q x_q, \tag{4.1}$$

and we denote by

$$Q = Q\left(N_1, \ldots, N_d\right) = \prod_{q=1}^{d} \left(\left[-\frac{N_q}{2}, \frac{N_q}{2} \right) \cap \mathbb{Z} \right) \tag{4.2}$$

the d-dimensional orthotope with side lengths $N_1, \ldots, N_d$ centered at the origin. The letter N (without subscripts) denotes the cardinality of Q, so that

$$N = |Q| = \prod_{q=1}^{d} N_q. \tag{4.3}$$

Our principal objects of study in this chapter are frequency-sparse band-limited signals $S : [0,1]^d \to \mathbb{C}$ of the form

$$S(\boldsymbol{x}) = \sum_{j=1}^{k} a_j \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\omega}_j \cdot \boldsymbol{x}}, \tag{4.4}$$

where $a_j \in \mathbb{C}$. In this setting, "band-limited" means that $\boldsymbol{\omega}_j \in Q\left(N_1, \ldots, N_d\right)$ for some integers $N_1, \ldots, N_d$, and "frequency-sparse" means that $k \ll N = |Q|$. The d-dimensional Fourier series $\widehat{S}(\boldsymbol{\omega})$ of a signal $S(\boldsymbol{x})$ is defined analogously to (2.2) by

$$\widehat{S}(\boldsymbol{\omega}) = \int_{[0,1]^d} S(\boldsymbol{x}) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\omega} \cdot \boldsymbol{x}} \mathrm{d}\boldsymbol{x}, \ \boldsymbol{\omega} \in \mathbb{Z}^d. \tag{4.5}$$

For signals of the form (4.4) we have

$$\begin{aligned} \widehat{S}(\boldsymbol{\omega}_\ell) &= \int_{[0,1]^d} \left(\sum_{j=1}^{k} a_j \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\omega}_j \cdot \boldsymbol{x}} \right) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{\omega}_\ell \cdot \boldsymbol{x}} \mathrm{d}\boldsymbol{x} \\ &= \sum_{j=1}^{k} a_j \int_{[0,1]^d} \mathrm{e}^{2\pi \mathrm{i} (\boldsymbol{\omega}_j - \boldsymbol{\omega}_\ell) \cdot \boldsymbol{x}} \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{4.6}$$

The separability of the exponentials $\mathrm{e}^{\boldsymbol{x}} = \mathrm{e}^{x}1 \cdots \mathrm{e}^{x}d$ implies that the integral in (4.6) factors into the product of d one-dimensional integrals of the form

$$\int_0^1 \mathrm{e}^{2\pi \mathrm{i} (\omega_{j,q} - \omega_{\ell,q}) x_q} \mathrm{d}x_q, \ q = 1, \ldots, d. \tag{4.7}$$

Since all components of the frequency vectors $\boldsymbol{\omega}_j$ and $\boldsymbol{\omega}_\ell$ are integers, each of these integrals is just the Kronecker delta function $\delta_{j\ell}$, so that $\widehat{S}(\boldsymbol{\omega}_\ell) = a_\ell$ for $\ell = 1, \ldots, k$ and 0 for all other $\boldsymbol{\omega} \in Q$.

Given a finite d-dimensional array $\boldsymbol{S}$ of size $p_1 \times \cdots \times p_d$ we define its discrete Fourier

transform by

$$\widehat{\boldsymbol{S}}[\boldsymbol{h}] = \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \boldsymbol{S}[\boldsymbol{j}] \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{h}\cdot\boldsymbol{j}/\boldsymbol{p}}, \tag{4.8}$$

where $\boldsymbol{p} = (p_1, \ldots, p_d)$, the indices h_q and j_q range from 0 to $p_q - 1$ for $q = 1, \ldots, d$, the division $\boldsymbol{j}/\boldsymbol{p} \stackrel{\text{def}}{=} (j_1/p_1, \ldots, j_d/p_d)$ is defined elementwise, and the sum over the vector index $\boldsymbol{j}$ is nested and performed over each dimension, so that

$$\sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \stackrel{\text{def}}{=} \sum_{j_1=0}^{p_1-1} \cdots \sum_{j_d=0}^{p_d-1}. \tag{4.9}$$

For $p = p_1 \cdots p_d$, multi-dimensional FFT algorithms [46] permit the computation of $\widehat{\boldsymbol{S}}$ in $\mathrm{O}(p \log p)$ time.

As in chapter , the simplest method to determine the significant frequency/coefficient pairs $\left\{(\boldsymbol{\omega}_j, a_j)\right\}_{j=1}^{k}$ in the Fourier transform of a signal S of the form (4.4) is to sample at the Nyquist rate in each dimension and compute the full d-dimensional DFT. For a signal bandlimited to $Q(N_1, \ldots, N_d)$ this procedure has sampling and runtime complexities of $\Theta(N)$ and $\Theta(N \log N)$, respectively, where $N = |Q|$. When $k \ll N$ this is computationally wasteful, and in the following sections we develop methods which are sub-linear in N.

4.3 Straightforward extension to multiple dimensions

In this section we describe a straightforward extension of the algorithm given in chapter to multiple dimensions. Recall that in the one-dimensional case, we used the fact that the phase modulation in the DFT coefficients of time-shifted samples of the input is linear in the frequency index. In section 4.3.1 we show that the same is true in multiple dimensions.

In section 4.3.2 we exploit this fact to develop an algorithm with sampling and runtime complexities of $\Theta\left(k^d\right)$ and $\Theta\left(k^d \log\left(k^d\right)\right)$, respectively.

4.3.1 Preliminaries

As in chapter we apply the d-dimensional DFT to discrete samples of $S(\boldsymbol{x})$ to determine the significant frequency/coefficient pairs $\left\{\left(\boldsymbol{\omega}_j, a_j\right)\right\}_{j=1}^{k}$. Fix integers $p_1, \ldots, p_d$ and positive real numbers $\varepsilon_1, \ldots, \varepsilon_d$. We form $d+1$ discrete d-dimensional arrays $\boldsymbol{S}_{\boldsymbol{p}}, \boldsymbol{S}_{\boldsymbol{p},\varepsilon_1}, \ldots, \boldsymbol{S}_{\boldsymbol{p},\varepsilon_d}$ by sampling $S(\boldsymbol{x})$ at rate $1/p_q$ in the q^{th} dimension starting with $x_q = 0$ and $x_q = \varepsilon_q$. That is, for $0 \le j_q \le p_q - 1$,

$$\begin{aligned} \boldsymbol{S}_{\boldsymbol{p}}[j_1, \ldots, j_d] &= S\left(\frac{j_1}{p_1}, \ldots, \frac{j_d}{p_d}\right), \\ \boldsymbol{S}_{\boldsymbol{p},\varepsilon_q}[j_1, \ldots, j_d] &= S\left(\frac{j_1}{p_1}, \ldots, \frac{j_q}{p_q} + \varepsilon_q, \ldots, \frac{j_d}{p_d}\right). \end{aligned} \tag{4.10}$$

If $S(\boldsymbol{x})$ is of the form (4.4), we have (as in chapter)

$$\begin{aligned} \widehat{\boldsymbol{S}}_{\boldsymbol{p}}[\boldsymbol{h}] &= \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} S\left(\frac{\boldsymbol{j}}{\boldsymbol{p}}\right) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{h}\cdot\boldsymbol{j}/\boldsymbol{p}} \\ &= \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \left(\sum_{\ell=1}^{k} a_\ell \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\omega}_\ell \cdot \boldsymbol{j}/\boldsymbol{p}} \right) \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{h}\cdot\boldsymbol{j}/\boldsymbol{p}} \\ &= \sum_{\ell=1}^{k} a_\ell \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \mathrm{e}^{2\pi \mathrm{i} (\boldsymbol{\omega}_\ell - \boldsymbol{h})\cdot\boldsymbol{j}/\boldsymbol{p}} \\ &= p_1 \cdots p_d \sum_{\boldsymbol{\omega}_\ell \equiv \boldsymbol{h} \pmod{\boldsymbol{p}}} a_\ell. \end{aligned} \tag{4.11}$$

Here as before, the division of vectors $\boldsymbol{j}/\boldsymbol{p} = (j_1/p_1, \ldots, j_d/p_d)$ is to be interpreted ele-

mentwise, and $\boldsymbol{\omega}_\ell \equiv \boldsymbol{h} \pmod{\boldsymbol{p}}$ is shorthand for the d simultaneous congruences $\omega_{\ell,q} \equiv h_q \pmod{p_q},\ q = 1, \ldots, d$. The last equality in (4.11) can be seen as follows: in expanded form, the inner sum in the second-to-last line in (4.11) reads

$$\sum_{j_1=0}^{p_1-1} \cdots \sum_{j_d=0}^{p_d-1} \mathrm{e}^{2\pi\mathrm{i}\left[(\omega_{\ell,1}-h_1)j_1/p_1 + \cdots + (\omega_{\ell,d}-h_d)j_d/p_d\right]}, \tag{4.12}$$

which can be rewritten as the product of d sums

$$\prod_{q=1}^{d} \sum_{j_q=0}^{p_q-1} \mathrm{e}^{2\pi\mathrm{i}(\omega_{\ell,q}-h_q)j_q/p_q}. \tag{4.13}$$

As in chapter , each of the sums in (4.13) is equal to p_q if $(\omega_{\ell,q} - h_q)/p_q$ is integral, and zero otherwise. The product is thus equal to $p_1 \cdots p_d$ if $\omega_{\ell,q} \equiv h_q \pmod{p_q}$ for $1 \le q \le d$ and zero otherwise, hence (4.11).

Similarly, for the shifted samples we have

$$\begin{aligned}
\widehat{\boldsymbol{S}}_{\boldsymbol{p},\varepsilon_q}[\boldsymbol{h}] &= \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \boldsymbol{S}_{\boldsymbol{p},\varepsilon_q}[\boldsymbol{j}] \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\cdot\boldsymbol{j}/\boldsymbol{p}} \\
&= \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \left(\sum_{\ell=1}^{k} a_\ell \mathrm{e}^{2\pi\mathrm{i}\left(\boldsymbol{\omega}_\ell\cdot\boldsymbol{j}/\boldsymbol{p} + \omega_{\ell,q}\varepsilon q\right)} \right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{h}\cdot\boldsymbol{j}/\boldsymbol{p}} \\
&= \sum_{\ell=1}^{k} a_\ell \sum_{\boldsymbol{j}=0}^{\boldsymbol{p}-1} \mathrm{e}^{2\pi\mathrm{i}\left((\boldsymbol{\omega}_\ell-\boldsymbol{h})\cdot\boldsymbol{j}/\boldsymbol{p} + \omega_{\ell,q}\varepsilon q\right)} \\
&= p_1 \cdots p_d \sum_{\boldsymbol{\omega}_\ell \equiv \boldsymbol{h} \pmod{\boldsymbol{p}}} a_\ell \mathrm{e}^{2\pi\mathrm{i}\omega_{\ell,q}\varepsilon q}.
\end{aligned} \tag{4.14}$$

Assuming for the moment that $\{\boldsymbol{\omega}_\ell \pmod{\boldsymbol{p}}\}_{\ell=1}^{k}$ are distinct (as d-vectors), we then

have for the unshifted samples

$$\widehat{\boldsymbol{S}}_{\boldsymbol{p}}[\boldsymbol{h}] = \begin{cases} p_1 \cdots p_d a_\ell & \text{if } \boldsymbol{\omega}_\ell \equiv \boldsymbol{h} \pmod{\boldsymbol{p}}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.15}$$

while for the shifted samples we have, for $1 \leq q \leq d$,

$$\widehat{\boldsymbol{S}}_{\boldsymbol{p},\varepsilon q}[\boldsymbol{h}] = \begin{cases} p_1 \cdots p_d a_\ell \mathrm{e}^{2\pi \mathrm{i} \varepsilon q \omega_{\ell,q}} & \text{if } \boldsymbol{\omega}_\ell \equiv \boldsymbol{h} \pmod{\boldsymbol{p}}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

As in the one-dimensional case, if we take $\varepsilon_q \leq \frac{1}{N_q}$, we can recover $\omega_{\ell,q}$ by noting that, for $\boldsymbol{h} \equiv \boldsymbol{\omega}_\ell \pmod{\boldsymbol{p}}$,

$$\omega_{\ell,q} = \frac{1}{2\pi\varepsilon_q} \operatorname{Arg}\left(\frac{\widehat{\boldsymbol{S}}_{\boldsymbol{p},\varepsilon_q}[\boldsymbol{h}]}{\widehat{\boldsymbol{S}}_{\boldsymbol{p}}[\boldsymbol{h}]} \right). \tag{4.17}$$

4.3.2 Algorithm

Equation (4.17) allows us to quickly extract the significant frequency values in the multi-dimensional DFT of an input signal $S(\boldsymbol{x})$ provided that no aliasing occurs in a given (d-dimensional) frequency bin $\boldsymbol{h}$. Recall that in the one-dimensional case, we tested whether or not a collision had occurred in a frequency bin by comparing the magnitudes of the shifted and unshifted coefficients for that bin. Moreover, we took the sample lengths p_j to be prime numbers greater than a constant multiple of k (or k^* in the adaptive case). In section 2.4 we proved that with this choice of p_j, the expected number of collisions among k frequencies is a constant fraction of k, so that the computational cost of each iteration of algorithm 1 decreases geometrically in the iteration number. This in turn implied that the total computational cost is dominated by that of the first iteration, which gave us sampling

and runtime complexities of $\Theta(k)$ and $\Theta(k\log(k))$, respectively.

In multiple dimensions, collisions among frequencies can occur in any of the d dimensions. A naive approach is then to only calculate frequency values via (4.17) for those frequency bins $\boldsymbol{h}$ for which the ratio of shifted to unshifted coefficients is sufficiently close to unity *in all dimensions*. That is, for given tolerances τ_q, we require all of the inequalities

$$\left| \frac{\left| \widehat{\boldsymbol{S}}_{\boldsymbol{p},\varepsilon_q}[\boldsymbol{h}] \right|}{\left| \widehat{\boldsymbol{S}}_{\boldsymbol{p}}[\boldsymbol{h}] \right|} - 1 \right| \leq \tau_q,\ 1 \leq q \leq d, \tag{4.18}$$

to hold in order to accept the frequency bin $\boldsymbol{h}$. For the simulations described in section 4.5 we took $\tau_q = p_1 \cdots p_d / N_1 \cdots N_d$.

In order to apply the average-case analysis of section 2.4 we therefore require the sampling rate in *each* dimension to be at least a constant multiple of the sparsity – that is, we take $p_{j,q} \geq ck$ (or ck^*) for $1 \leq q \leq d$. As discussed in section 2.4.1, the computational costs of our algorithm are dominated by the FFTs and sorts of the sample arrays $\boldsymbol{S}_{\boldsymbol{p}}$ and $\boldsymbol{S}_{\boldsymbol{p},\varepsilon_q}$. The sampling complexity of this naive algorithm is then $\Theta\left(k^d\right)$, since we require that many signal values on the grid defined by the cartesian product of the $p_{j,q}$s. As the d-dimensional FFT of an array of size $\Theta(k) \times \cdots \times \Theta(k)$ has time complexity $\Theta\left(k^d \log\left(k^d\right)\right)$, this naive algorithm will execute in time $\Theta\left(k^d \log\left(k^d\right)\right)$ as well.

Remark 4.3.1. The empirical evaluation in section 4.5 indicates that this choice of $p_{j,q}$ values is overly cautious, and that a coarser sampling of the input may suffice for energetic frequency identification. Specifically, it was observed that the naive algorithm described above almost never detected an aliased frequency bin, so that on average only one set of sample rates $\boldsymbol{p}$ was used. The explanation for this observation is that by choosing the sample

rates $p_{j,q} \geq ck$, we are effectively hashing the k frequencies into $\mathrm{O}\left(k^d\right)$ d-dimensional bins, since the discrete grid defined by the cartesian product of the $p_{j,q}$s has $(ck)^d$ nodes. For $d > 1$ we should therefore expect many fewer multiple-occupancy d-dimensional bins than in the $d = 1$ case.

We therefore tested a variant of the naive algorithm in which we take $p_{j,q} \geq (ck)^{1/d}$ (or $(ck^*)^{1/d}$) for $q = 1, \ldots, d$. In this manner the *total* number of d-dimensional bins is ck (or ck^*), so the sampling and runtime complexities of one iteration of the inner loop are $\Theta(k)$ and $\Theta(k \log k)$, respectively. On average a constant fraction of the frequencies end up in single-occupancy d-dimensional frequency bins, which by (4.14) is sufficient for calculating the frequency index. Since we test for aliasing in every dimension, we would expect this variant of the naive algorithm to reject more frequencies than our one-dimensional algorithm. The results of the empirical evaluation support this observation.

4.4 Signal unwrapping and the Chinese Remainder Theorem

In this section we provide an alternative to the naive algorithm described above that avoids the exponential scaling of the sampling and runtime complexities in the dimension d. We achieve this by using a clever number-theoretic technique to "unwrap" the d-dimensional input signal $S(\boldsymbol{x})$ into a related one-dimensional signal $S^{\mathrm{u}}(x)$. We then apply the one-dimensional algorithm of chapter and transform back to d-dimensional indices. As we will show in section 4.4.1, this dimension-reduction technique follows from the Chinese Remainder Theorem (Theorem 2.2.5), which we used previously for worst-case bounds on the number

of $p_j s$ used by the one-dimensional algorithm.

4.4.1 Preliminaries

In this section we provide the details of the dimension-reducing unwrapping technique, based on ideas presented in [29] and [7]. Let $S(\boldsymbol{x})$ be a signal of the form (4.4), and let $Q = Q\left(N_1, \ldots, N_d\right)$ be as in (4.2) so that $\text{supp}\left(\widehat{S}\right) \subset Q$. Next, choose d relatively prime integers $\widetilde{N}_q,\ 1 \leq q \leq d$, such that

$$\widetilde{N}_q \geq dN_q, \tag{4.19}$$

and set

$$\widetilde{N} = \prod_{q=1}^{d} \widetilde{N}_q. \tag{4.20}$$

Finally, let $y^{-1} \pmod{\widetilde{N}_q}$ denote the multiplicative inverse of y in $\mathbb{Z}_{\widetilde{N}_q}$, so that $y \cdot y^{-1} \pmod{\widetilde{N}_q} \equiv 1 \pmod{\widetilde{N}_q}$. Note that $y^{-1} \pmod{\widetilde{N}_q}$ exists, and is unique, whenever y is relatively prime to $\widetilde{N}_q$.

Define the function $\iota : Q\left(N_1, \ldots, N_d\right) \to \mathbb{Z}_{\widetilde{N}}$ by

$$\iota(\boldsymbol{x}) = \left(\sum_{q=1}^{d} \frac{\widetilde{N}}{\widetilde{N}_q} x_q \right) \pmod{\widetilde{N}}. \tag{4.21}$$

Since the $\widetilde{N}_q$ are coprime, the Chinese Remainder Theorem implies that ι is a bijection, whose inverse $\boldsymbol{x} = \iota^{-1}(x)$ has components

$$x_q = x \left(\frac{\widetilde{N}}{\widetilde{N}_q} \right)^{-1 \pmod{\widetilde{N}_q}}. \tag{4.22}$$

Given these parameters, we can define a one-dimensional function $S^{\mathrm{u}} : [0,1] \to \mathbb{C}$ (the "unwrapped" version of S) by

$$S^{\mathrm{u}}(x) = S\left(\frac{\widetilde{N}}{\widetilde{N}_1}x, \ldots, \frac{\widetilde{N}}{\widetilde{N}_d}x\right). \tag{4.23}$$

For notational convenience in (4.24) below, let $\boldsymbol{x}$ denote the vector argument of S in (4.23). If S is of the form (4.4), we then have

$$\begin{aligned}
\widehat{S}^{\mathrm{u}}(\omega) &= \int_0^1 S^{\mathrm{u}}(x)\mathrm{e}^{-2\pi\mathrm{i}\omega x}\mathrm{d}x \\
&= \int_0^1 \left(\sum_{j=1}^{k} \widehat{S}(\boldsymbol{\omega}_j)\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{\omega}_j\cdot\boldsymbol{x}}\right)\mathrm{e}^{-2\pi\mathrm{i}\omega x}\mathrm{d}x \\
&= \sum_{j=1}^{k} \widehat{S}(\boldsymbol{\omega}_j)\int_0^1 \exp\left(-2\pi\mathrm{i}x\left(\omega - \sum_{q=1}^{d}\frac{\widetilde{N}}{\widetilde{N}_q}\omega_{j,q}\right)\right)\mathrm{d}x.
\end{aligned} \tag{4.24}$$

From (4.24) we see that $\widehat{S}^{\mathrm{u}}$ is non-zero only at the frequencies

$$\omega_j = \sum_{q=1}^{d}\frac{\widetilde{N}}{\widetilde{N}_q}\omega_{j,q} = \iota(\boldsymbol{\omega}_j), \tag{4.25}$$

and moreover we have $\widehat{S}^{\mathrm{u}}(\omega) = \widehat{S}\left(\iota^{-1}(\omega)\right)$.

Finally, we note that if $\widehat{S}(\boldsymbol{\omega})$ is band-limited to Q, then $\widehat{S}^{\mathrm{u}}(\omega)$ is band-limited to $\left[-\frac{\widetilde{N}}{2}, \frac{\widetilde{N}}{2}\right)$. To see this, note that (4.25) implies

$$|\omega| \leq \sum_{q=1}^{d}\left|\frac{\widetilde{N}}{\widetilde{N}_q}\omega_q\right| \leq \sum_{q=1}^{d}\frac{\widetilde{N}}{\widetilde{N}_q}\frac{N_q}{2}$$

$$\leq \sum_{q=1}^{d} \frac{\widetilde{N}}{2d} = \frac{\widetilde{N}}{2}, \tag{4.26}$$

where we used the band-limitedness of ω_q to $\left[-\frac{\widetilde{N}_q}{2}, \frac{\widetilde{N}_q}{2}\right)$ in the second inequality and the choice of $\widetilde{N}_q$ from (4.19) in the third. Thus $S^{\mathrm{u}}(x)$ is frequency-sparse and band-limited whenever $S(\boldsymbol{x})$ is, so we may apply all the results of chapter to obtain a multi-dimensional algorithm.

4.4.2 Algorithm

Using the bijective unwrapping ι from (4.21), we have a simple prescription for a multi-dimensional algorithm. Given Q we first precompute $\widetilde{N}_q$, $\widetilde{N}$, and $\left(\widetilde{N}/\widetilde{N}_q\right)^{-1} \pmod{\widetilde{N}_q}$ for $1 \leq q \leq d$ using the extended Euclidean algorithm [13]. This allows us to evaluate S^{u} at any desired point $\boldsymbol{x}$ using (4.23). We then use the one-dimensional algorithm of chapter to approximate $\widehat{S}^{\mathrm{u}}$, and finally transform back to d dimensions using ι^{-1}. The sampling and runtime complexities of this "unwrapped" algorithm are $\Theta(k)$ and $\Theta(k \log k)$, respectively.

Remark 4.4.1. Since this "unwrapped" algorithm merely invokes the one-dimensional algorithm of chapter , we would expect it to perform similarly to that algorithm in practice. While it does seem to attain the stated sampling complexity, the results of the empirical evaluation in section 4.5 imply that it performs most similarly to the variation on the naive algorithm mentioned in remark 4.3.1. This is somewhat surprising, and we leave the relationship between the two algorithms as future work.

4.5 Empirical evaluation

In this section we describe the results of an empirical evaluation in two dimensions of the three multi-dimensional algorithms described in sections 4.3 and 4.4. We only tested the *adaptive* versions of these algorithms, where the sample lengths are chosen with respect to k^*, the number of frequencies left in the residual. As in chapter , all three variants were implemented in C++ using FFTW 3.0 [19] for the FFTs, and all code was compiled with the Intel compiler using the `-fast` optimization. The experiments were run on the same machine used in that section. In the figures below we refer to the three algorithms as "Naive" (section 4.3), "Naive-Sqrt" (section 4.3, remark 4.3.1), and "Unwrapped" (section 4.4).

4.5.1 Setup

For all of the experiments below, we fixed the bandwidth in both dimensions to be 2^{11}, so in the notation of section 4.2 we have $Q = [-1024, 1023) \times [-1024, 1023)$. This value was chosen to facilitate comparison with the one-dimensional results in section 2.5, where the (one-dimensional) bandwidth was taken to be 2^{22}. We varied the sparsity k from 2 to 512 by powers of two. All of the experiments were performed in the absence of noise, and we fixed the parameter $c_1 = 5$. For the Naive algorithm and its variant Naive-Sqrt we took the shifts $\varepsilon_q = 1/2N_q$, while for the Unwrapped version we took $\varepsilon = 1/2\widetilde{N}$. The test signals were of the form (4.4), and were generated by drawing frequencies $\boldsymbol{\omega}_j$ uniformly from Q and coefficients a_j uniformly from the complex unit circle.

Each data point in figures 4.1 and 4.2 represents the average over 100 independent trials, and the upper and lower bars in figure 4.1 represent the maximum and minimum values over all 100 trials. All variants of the algorithms recovered each test signal exactly (within 10^{-12}

in the ℓ_2 norm). Finally, we also compared our algorithm with the two-dimensional FFTW routine using `FFTW_MEASURE` plans.

4.5.2 Runtime and sampling complexity

In figure 4.1(a) we compare the average number of samples of the input $S(\boldsymbol{x})$ required by each algorithm when the bandwidth Q is fixed. We can see that the Naive algorithm scales approximately like $\Theta\left(k^2\right)$, while the Naive-Sqrt and Unwrapped algorithms scale apprimately like $\Theta(k)$, as expected. Note that for $k \gtrsim 256$, the Naive algorithm samples the input over the Nyquist rate $1/\left(N_1 N_2\right)$, while for all values of k tested the Naive-Sqrt and Unwrapped algorithms take orders of magnitude fewer samples than either FFTW or the Naive algorithm.

In figure 4.1(b) we compare the average runtime (in CPU ticks) of each algorithm when the bandwidth Q is fixed. The Naive algorithm is faster than FFTW for $k \leq 64$, while the Naive-Sqrt and Unwrapped algorithms are faster for $k \leq 256$. We can also see that the Naive algorithm scales approximately like $\Theta\left(k^2 \log(k)\right)$, and while the Naive-Sqrt and Unwrapped algorithms are generally two orders of magnitude faster, they also appear to scale at the same rate. This is in contrast to the expected $\Theta(k \log k)$ runtime complexity.

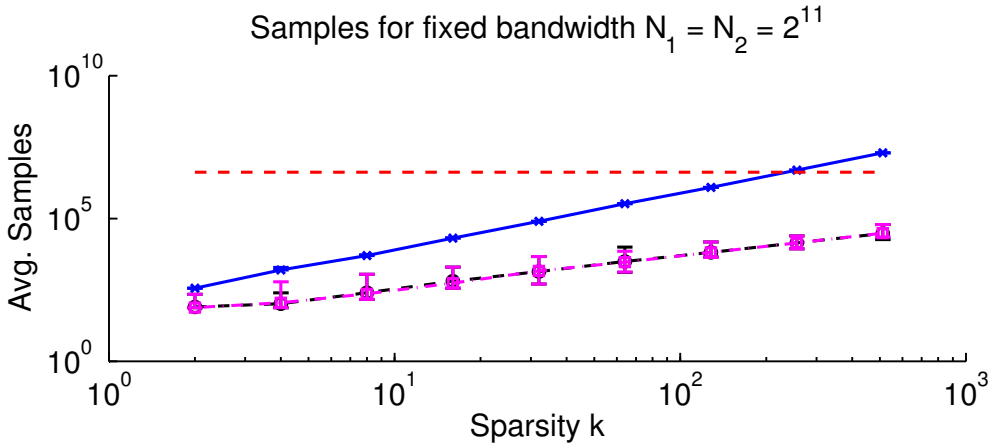
A potential explanation for this observation is that, as in the one-dimensional case, we evaluated the intermediate representations by simply looping over the sample points and the terms in the representation. As mentioned in section 2.4.1, this has time complexity $\Theta(k^2)$, while non-uniform FFTs can be used to reduce this to $\Theta(k \log k)$. Whereas in the one-dimensional case this did not affect the overall runtime, it appears that in two dimensions it represents a significant fraction of the runtime. We leave the replacement of the direct

evaluation with non-uniform FFTs in both the one- and multi-dimensional algorithms in this dissertation as future work.

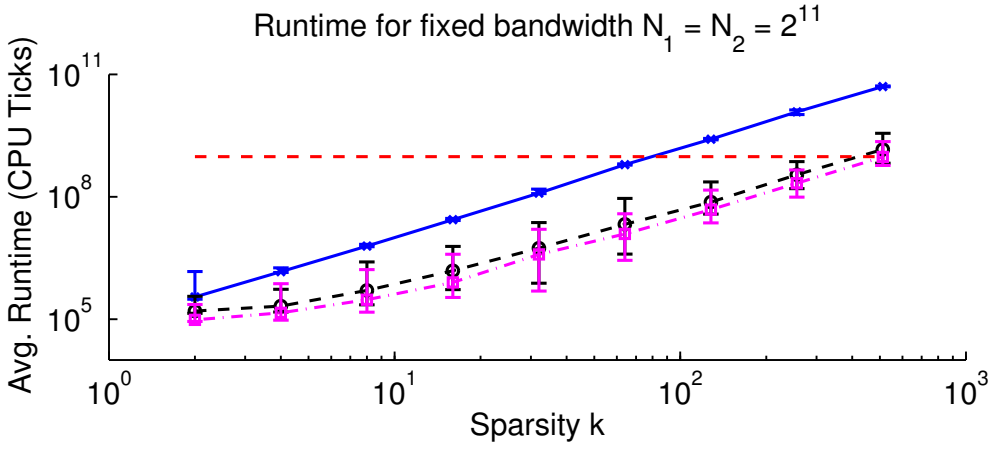
4.5.3 Sample lengths and aliased frequencies

In figure 4.2(a) we compare the average number of sets of sample lengths $\boldsymbol{p}_\ell$ used by each algorithm in the fixed bandwidth case. We include the one-dimensional algorithm with $N = 2^{22}$ for comparison. We can see that the Naive algorithm almost always uses only one set of sample lengths, which implies that the choice of $p_{\ell,q} \geq ck^*$ for $1 \leq q \leq d$ is overly cautious. When we reduce the number of frequency bins by only requiring $p_{\ell,q} \geq \sqrt{ck^*}$, the Naive-Sqrt algorithm uses many more sample length sets. Moreover, we see that the Unwrapped algorithm uses approximately the same number of sample lengths as the Naive-Sqrt algorithm, which is unexpected. What is more, both the Naive-Sqrt and Unwrapped algorithms use many more sample lengths than the one-dimensional algorithm on an equivalent bandwidth.

In figure 4.2(b) we compare the average fraction of aliased frequencies encountered by each algorithm over its execution. This quantity was calculated simply by incrementing a counter whenever the aliasing test (4.18) (or its equivalent in the Unwrapped and one-dimensional cases) failed, and then dividing the total number of failures by the sparsity k. Note that by the Markov chain analysis of one-dimensional algorithm in section 2.4.3 this quantity should be independent of k, and this is indeed observed for the one-dimensional algorithm. As mentioned in remark 4.3.1, we can see that the Naive algorithm almost never encounters an aliased frequency, while the Naive-Sqrt and Unwrapped algorithms reject many more frequencies than in the equivalent one-dimensional case.

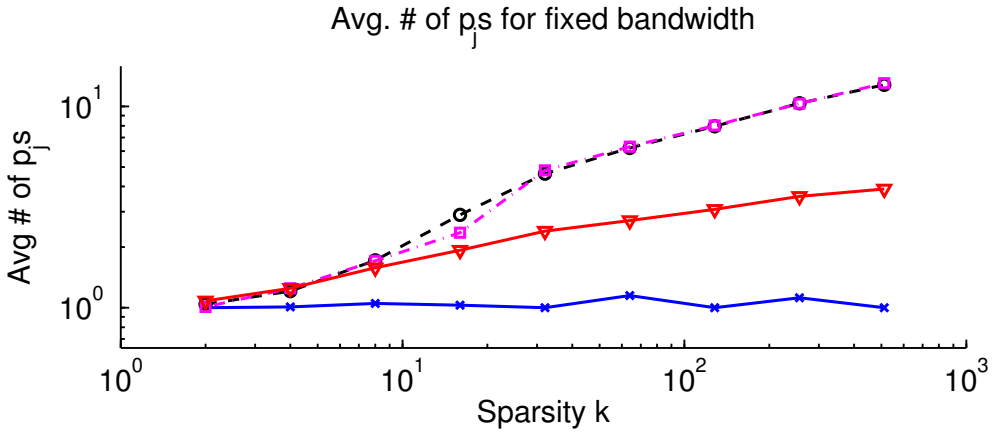


(a)

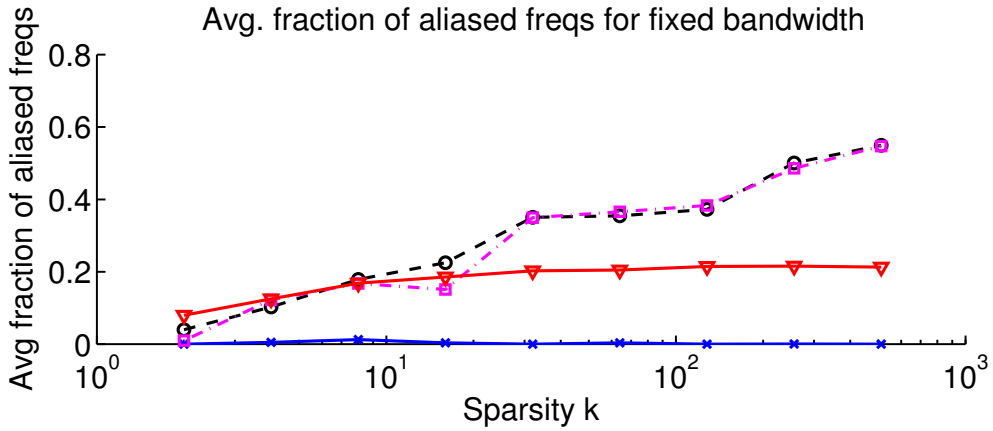


(b)

Figure 4.1: (a) 2D sampling complexity with fixed bandwidth $N_1 = N_2 = 2^{11}$ for Naive (blue solid line), Naive-Sqrt (black dashed line), Unwrapped (magenta dashed line), and FFTW (red dashed line). (b) 2D runtime complexity with fixed bandwidth $N_1 = N_2 = 2^{11}$ for Naive (blue solid line), Naive-Sqrt (black dashed line), Unwrapped (magenta dashed line), and FFTW (red dashed line).



(a)



(b)

Figure 4.2: (a) Average number of sample lengths used in 2D with fixed bandwidth $N_1 = N_2 = 2^{11}$ for Naive (solid blue line), Naive-Sqrt (black dashed line), and Unwrapped (magenta dashed line). Also shown for comparison is 1D PS-Det (red solid line) with fixed bandwidth $N = 2^{22}$. (b) Average fraction of aliased frequencies in 2D with fixed bandwidth $N_1 = N_2 = 2^{11}$ for Naive (solid blue line), Naive-Sqrt (black dashed line), and Unwrapped (magenta dashed line). Also shown for comparison is 1D PS-Det (red solid line) with fixed bandwidth $N = 2^{22}$.

A possible explanation for the increased number of sample length sets $\boldsymbol{p}_\ell$ and the increased fraction of aliased frequencies observed in the Naive-Sqrt algorithm is the test used for aliasing in a given frequency bin. In the one-dimensional case, we tested whether the ratio between the magnitudes of the shifted and unshifted peaks was sufficiently close to unity, whereas in the multi-dimensional case we require that a frequency bin pass this test in *all* dimensions. The requirements for accepting a frequency bin are therefore more stringent in higher dimensions, and we would expect a commensurate increase in the number of rejected frequencies. As shown in figure 4.2 this is indeed observed in two dimensions for the Naive-Sqrt algorithm.

The effect of this increase in the number of rejected frequencies can also be seen in the runtime of the two-dimensional Naive-Sqrt algorithm. As more frequencies are rejected, more sample lengths $\boldsymbol{p}_\ell$ are used to identify the remaining frequencies. Recall that in the adaptive algorithm (which is what we implemented), whenever a new sample rate is used we subtract off the contribution of the already-discovered Fourier terms. Since our implementation currently performs this operation directly, with time complexity $\Theta\left(k^2\right)$, the increased number of rejected frequencies leads to the overall $\Theta\left(k^2\right)$ time complexity observed in figure 4.1(b).

Finally we note that the preceding discussion only seems to apply to the Naive-Sqrt algorithm, and not the Unwrapped version. The aliasing test in the Unwrapped algorithm is identical to the one-dimensional version, so we would not expect an increase in the number of rejected frequencies. The fact that the Unwrapped algorithm has near-identical empirical performance with the Naive-Sqrt algorithm remains somewhat mysterious, and we leave the elucidation of the relationship between the two as future work.

4.6 Conclusion

In this chapter we gave three different extensions of our basic algorithm to the multi-dimensional setting. Two of these rely on the separability properties of the exponential function, while the third relies on a clever number-theoretic construction using the Chinese Remainder Theorem. The resulting algorithms have runtime and sampling requirements that are less stringent than the d-dimensional FFT for sparsity levels up to $k = 256$. We remark again that the implementation of non-uniform FFTs in place of direct evaluation of intermediate representations and the explanation of the similarity in the observed behavior of the Naive-Sqrt and Unwrapped algorithms are left as possible directions for future research.

BIBLIOGRAPHY

BIBLIOGRAPHY

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

[2] A. Akavia, *Deterministic Sparse Fourier Approximation via Fooling Arithmetic Progressions*, Conference on Learning Theory (CoLT), 2010.

[3] 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.

[4] C. Anderson and M. D. Dahleh, *Rapid computation of the discrete Fourier transform*, SIAM J. Sci. Comput. **17** (1996), no. 4, 913–919.

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

[6] W. L. Briggs and V. E. Henson, *The DFT: an owner's manual for the discrete Fourier transform*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995.

[7] C. Burrus, *Index mappings for multidimensional formulation of the DFT and convolution*, IEEE Transactions on Acoustics, Speech and Signal Processing **25** (1977), no. 3, 239–242.

[8] E. J. Candès, J. K. Romberg, and T. Tao, *Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory **52** (2006), no. 2, 489–509.

[9] I. Carron, *Nuit blanche*, `http://nuit-blanche.blogspot.com`.

[10] S.S. Chen, D.L. Donoho, and M.A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput. **20** (1998), no. 1, 33–61.

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

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

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, second ed., MIT Press, Cambridge, MA, 2001.

[14] L. Demanet and G. Peyré, *Compressive wave computation*, Submitted to Found. Comput. Math. (2008), Preprint available at `http://math.mit.edu/ laurent/html/publications.html`.

[15] J. Dongarra and F. Sullivan, *Guest editors' introduction: The top 10 algorithms*, Computing in Science and Engineering (2000), 22–23.

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

[17] D. P. Dubhashi and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*, Cambridge University Press, Cambridge, 2009.

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

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

[20] 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.

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

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

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

[24] G. Golub and C. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.

[25] 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.

[26] ———, *Simple and practical algorithms for sparse Fourier transform*, to appear in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2012.

[27] M. Iwen, *A deterministic sub-linear time sparse Fourier algorithm via non-adaptive compressed sensing methods*, Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008, pp. 20–29.

[28] ———, *Combinatorial sublinear-time Fourier algorithms*, Found. Comput. Math. **10** (2010), no. 3, 303–338.

[29] ———, *Improved approximation guarantees for sublinear-time Fourier algorithms*, Submitted (2011), Available at `http://www.math.duke.edu/ markiwen/`.

[30] 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.

[31] R. M. Karp, *Probabilistic recurrence relations*, J. Assoc. Comput. Mach. **41** (1994), no. 6, 1136–1150.

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

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

[34] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Math. Oper. Res. **8** (1983), no. 4, 538–548.

[35] T. Lin and F.J. Herrmann, *Compressed wavefield extrapolation*, Geophysics **72** (2007), no. 5, SM77–SM93.

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

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

[38] I. Niven, H.S. Zuckerman, and H.L. Montgomery, *An introduction to the theory of numbers*, fifth ed., John Wiley & Sons Inc., New York, 1991.

[39] A.M. Pinkus, *On L^1-approximation*, Cambridge Tracts in Mathematics, vol. 93, Cambridge University Press, Cambridge, 1989.

[40] Y. Rubner, C. Tomasi, and L.J. Guibas, *The earth mover's distance as a metric for image retrieval*, International Journal of Computer Vision **40** (2000), no. 2, 99–121.

[41] L.I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: Nonlinear Phenomena **60** (1992), no. 1-4, 259–268.

[42] F. Santosa and W.W. Symes, *Linear inversion of band-limited reflection seismograms*, SIAM J. Sci. Statist. Comput. **7** (1986), no. 4, 1307–1330.

[43] A. Schrijver, *Theory of linear and integer programming*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons Ltd., Chichester, 1986.

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

[45] T. Tao and V. Vu, *Additive combinatorics*, Cambridge Studies in Advanced Mathematics, vol. 105, Cambridge University Press, Cambridge, 2006.

[46] C. Van Loan, *Computational frameworks for the fast Fourier transform*, Frontiers in Applied Mathematics, vol. 10, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

[47] Y. Wang and G. Zhou, *On the use of high-order ambiguity function for multi-component polynomial phase signals*, Signal Processing **65** (1998), no. 2, 283–296.