# SPARSE HARMONIC TRANSFORMS

By

Bosu Choi

A DISSERTATION

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

Applied Mathematics — Doctor of Philosophy

2018

**ABSTRACT**

SPARSE HARMONIC TRANSFORMS

By

Bosu Choi

We develop fast function learning algorithms given an assumption that a function can be well approximated by the expansion of a few high-dimensional basis functions. Considering the tensorized Fourier basis functions, several versions of high-dimensional sparse Fourier transforms (SFTs) are discussed. One-dimensional sparse Fourier transforms introduced in [1] and [2] quickly approximate functions represented by only $s$ Fourier basis functions using a few samples (function evaluations) without noise and with noise respectively. The algorithms can be directly applied to compute the Fourier transform of the high-dimensional functions. However, it becomes hard to implement them if the dimension gets too large. In this thesis, we introduce two new concepts: *partial unwrapping* and *tilting*. These two ideas allow us to efficiently compute the high-dimensional sparse Fourier transforms using the ideas in [1] and [2]. Furthermore, we develop sublinear-time compressive sensing methods to approximate the multivariate functions by the expansion of a few bounded orthonormal product (BOP) bases which include tensorized Fourier basis functions. These new methods are obtained from CoSaMP by replacing its usual support identification procedure with a new faster one inspired by fast SFT techniques. The resulting sublinearized CoSaMP method allows for the rapid approximation of bounded orthonormal product basis (BOPB)-sparse functions of many variables which are too hideously high-dimensional to be learned by other means. Both numerics and theoretical recovery guarantees will be presented.

## ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisors, professors Andrew Christlieb and Mark Iwen for their continuous support and encouragement during the six years of my Ph.D study. They have provided me with the tremendous knowledge, motivation, great insight, patient guidance, and supportive research environment. I also thank professor Yang Wang at HKUST, Hong Kong and professor Felix Krahmer at TUM, Germany who are my valuable collaborators giving me brilliant research ideas and chance to visit their institutes. I have been extremely lucky to have the best advisors and collaborators for my Ph.D. study so that I could finish my Ph.D research and have a chance to continue my research career after the graduation.

I also thank my committee members, professor Yingda Cheng, for teaching me great knowledge in numerical analysis, professor Matthew Hirn for his valuable feedback and comments on my thesis, and professor Zhengfang Zhou for his enlightening opinion for the future research extension. I also thank my Korean advisor, professor Jungho Yoon, for his continuous guidance and support for being a Ph.D and a researcher.

I thank all my dear friends at MSU, EWHA Univ. and EWHA High. I am so lucky to have so many good friends who supported me whether we are in remote distance or not. Especially, I want to thank my friends Dr. Hana Cho and Dr. Hyosun Yang with whom I spent most of my 20's to pursue a math bachelor's, master's, and Ph.D degrees. Thanks to them, I have not been alone, struggling to do research and be a mathematician eventually since professor Jungho Yoon inspired us in our freshman year in EWHA Univ.

I am eternally grateful to my family: my parents, my brother Boksoo, and my sister Heechan for their unconditional support. I also thank my grandparents, relatives and cousins, both in South Korea and the United States, who gave me courage to pursue a Ph.D degree with special gatherings, so I could enjoy being away from my Ph.D study occasionally.

# TABLE OF CONTENTS

# LIST OF FIGURES

# Chapter 1

# Introduction

In this thesis, we develop sublinear-time algorithms for finding signals under the assumption of sparsity. We first do this in the Fourier setting and then develop a generalized method which is expected to be applied to Uncertainty Quantification(UQ) with great promise.

In various numerical problems, one encounters the problem of function approximation, interpolation, and learning from finite function evaluations (samples). For high-dimensional function domains, there is the danger of the curse of dimensionality, i.e., the problem that the number of function evaluations required for an accurate result grows exponentially fast as the dimension grows. For example, one of the best known and most frequently-used algorithms is the Fast Fourier Transforms (FFT). However, in the case that the bandwidth $M$ of the frequencies is large, the sampling size becomes large, as dictated by the Shannon-Nyquist sampling theorem. Specifically, the runtime complexity is $\mathcal{O}(M\log M)$ and the number of samples is $\mathcal{O}(M)$. This issue is only exacerbated in the $D$-dimensional setting, where the runtime complexity is $\mathcal{O}(M^D\log M^D)$ and the number of samples is $\mathcal{O}(M^D)$ if we assume the dimension is $D$ and the bandwidth in each dimension is $M$. For another example, it is proven in [3] that the numerical integration of $D$-dimensional functions in $C^r$ requires more than $c_r(1+\gamma)^D$ function evaluations for $\gamma > 0$ in order to achieve a given error $\epsilon$. Thus, some additional assumptions are endowed to the functions for developing a feasible algorithm. Sparsity is the one such assumption that we are often able to observe in many problems. That is, many signals or data show some sparse (or compressible) property in various forms.

Due to this curse of dimensionality, many higher dimensional problems of interest are beyond current computational capabilities of the traditional FFT. Moreover, in the sparse setting where the number $s$ of significant frequencies is small, it is computationally wasteful to compute all $M^D$ coefficients. In such a setting we refer to the problem as being sparse in the Fourier domain. On the other hand, for such sparse problems, the idea of sublinear sparse Fourier transforms was introduced in [4], and several sublinear algorithms were developed extending the idea [4, 5, 6, 7, 8, 1, 2]. These methods greatly reduce the runtime and sampling complexity of the FFT in the sparse setting. The methods were primarily designed for the one dimensional setting.

In [9], practical algorithms for data in two dimensions were given for the first time. In this thesis, we develop algorithms designed for higher dimensional data, which is effective even for dimensions in the hundreds and thousands. To achieve our goal, our approach must address the worst case scenario presented in [9]. It is not straightforward to extend one dimensional sparse Fourier transform algorithms to multiple dimensions. We face several obstacles. First, we do not have an efficient FFT for multidimensional problems much higher than three. Using projections onto lower-dimensional spaces solves this problem. However, like all projection methods for sparse FFT, one needs to match frequencies from one projection with those from another projection. This *registration problem* is one of the big challenges in the one dimensional sparse FFT. An equally difficult challenge is that different frequencies may be projected into the same frequency (*the collision problem*). All projection methods for sparse FFT primarily aim to overcome these two challenges. In higher dimensional sparse FFT, these problems become even more challenging as now we are dealing with frequency vectors, not just scalar frequencies.

In order to make our multi-dimensional sparse Fourier algorithm to be robust to noisy samples, we utilize the idea of a multiscale method from [2]. Suppose $f : [0,1)^D \to \mathbb{C}$ defined as $\sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{x}} + z(\boldsymbol{x})$ where $\boldsymbol{x} \in [0,1)^D$, $\boldsymbol{n} \in [-M/2, M/2)^D \cap \mathbb{Z}^D$, $|\mathcal{S}| = s \ll M^D$ and $z(\boldsymbol{x})$ represents a noise function. Our algorithms introduced in Chapter 2 are not robust to noise. This lack of robustness is due to the computation of the ratio between the discrete Fourier transforms of two sample sets from $f$ at the shifted and unshifted points in order to get the traces of significant Fourier modes. For example, consider a one mode sparse function $f(x) = ce^{2\pi i n x}$ of one dimension and two sample sets of length $p$, where $p$ is taken to be 1. That is, $\boldsymbol{f}_0 = (f(0))$ and $\boldsymbol{f}_\epsilon = (f(\epsilon))$ each of whose DFT yields the exact recovery of the frequency $n = \frac{1}{2\pi\epsilon}\text{Arg}\left(\frac{f(\epsilon)}{f(0)}\right)$ where the function Arg gives the argument falling into $[-\pi, \pi)$. If those samples are noisy, however, then $\frac{1}{2\pi\epsilon}\text{Arg}\left(\frac{f(\epsilon)+z(\epsilon)}{f(0)+z(0)}\right)$ does not guarantee the proper approximation of $n$. The fraction in $\text{Arg}(\cdot)$ in general is corrupted by the term whose size is $\mathcal{O}(\sigma/c_{\min}\sqrt{p})$ where $p$ is the number of samples, $c_{\min}$ is the nonzero Fourier coefficient least in magnitude, and the noise is assumed to be Gaussian noise $\sim \mathcal{N}(0, \sigma^2)$, details are given in Section 3.1.2. A significantly large $p > \mathcal{O}(\sigma/\epsilon c_{\min})^{2/3}$ enables us to find the correct Fourier modes by simple rounding as introduced in [2]. However, when either $\sigma$ is too big or the shift size $\epsilon$ is too small, $p$ is required to be very large, accordingly. In order to avoid $p$ excessively enlarged, we make use of a multiscale method in Chapter 3 which makes an initial guess about each frequency using moderately large $\epsilon_0$ and $p$, and add the finite $L$ correction terms computed by using gradually increasing $\epsilon_q$ for $q = 1, 2, \cdots, L$.

Our algorithms assume that we have access to an underlying continuous function $f$, i.e., we can sample at anywhere we want. However, samples are sometimes given at the beginning and getting extra samples can be very expensive. Accordingly, it is necessary to do the approximation of extra samples using given samples. A fully discrete sparse Fourier transform was introduced in [10] combining periodized Gaussian filters and one-dimensional sparse Fourier transform in the continuous setting such as [11] and [2]. We expect for the future work that this approach will help our multiscale high-dimensional algorithm to be modified to work for fully discrete samples.

Many problems are best represented using bases other than Fourier. In this thesis we develop a generalization of sublinear-time sparse harmonic transforms for such applications. To motivate our generalized sparse harmonic transform we will now consider an example from uncertainty quantification (UQ). A common class of examples in UQ literature [12, 13], is that certain quantities of interest (QoI) are affected by a number of parameters and one can assume that the QoI depend smoothly on these parameters. Consequently, uncertainty in these inputs affects the uncertainty in the QoI outputs. In the process of quantifying these uncertainties, the probability density function (PDF) of the input parameters is estimated through Bayesian inference, and the uncertainty is propagated through the models for sampling points chosen from the PDF in order to obtain the prediction intervals of QoI. A crucial step in this process is then to approximate the QoI from its sample values. This typically requires multivariate function integration and interpolation, usually via quadrature methods, sparse grid approaches, or Monte Carlo methods, depending on the number of parameters.

The first two methods indeed face the curse of dimensionality: Quadrature methods [14] require $\mathcal{O}(M^D)$ samples when $D$ is the number of dimensions/function variables. Sparse grid methods [15] improve on this by working with function spaces of mixed smoothness. Instead of a full grid using $\mathcal{O}(M^D)$ samples, they require only $\mathcal{O}(M(\log M)^{D-1})$ samples. Although it is better than quadrature methods for high dimensional problems, it still exhibits exponential dependence on $D$, and so they are only feasible for moderately large dimensional problems. Monte Carlo methods [16] on the other hand have the advantage of a convergence rate independent of the number of dimensions. However, they exhibit slow convergence at a rate of $\mathcal{O}(1/\sqrt{m})$, where $m$ is the sample number. Consequently, such methods are often used to integrate relatively high-dimensional functions. Similarly, quasi-Monte Carlo[17, 16] achieves a convergence rate, $\mathcal{O}(\frac{(\log m)^D}{m})$ by using low-discrepancy sample sets. Though good for integration, (quasi-)Monte Carlo methods can not be used for function learning, approximation or interpolation. For such applications one has few options for large dimensions.

Uncertainty quantification consists of several stages. We first get several experimental observations of some phenomenon, design our model from observations using ODE, PDE, etc., and solve this model using various numerical methods. These three steps are revisited for validation and calibration. Each step contains uncertainty (error), and these errors are accumulated to uncertainty in QoI. Our models usually contain tens of or hundreds of parameters which are unknown. We consider parametric operator equations [18, 19, 20],

$$A(\boldsymbol{x})u(\boldsymbol{x}) = b,$$

where the solution $u(\boldsymbol{x})$ is in the Sobolev space $H_0^1(\mathcal{U})$, a parameter vector $\boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^D$ with a large $D \in \mathbb{N}$ and a probability measure $\boldsymbol{\nu}$ on $\mathcal{D}$, and $\mathcal{U}$ is the physical domain of $u$. QoI is approximated by a function $g : \boldsymbol{x} \to G(u(\boldsymbol{x}))$, where $G$ is a functional on $u \in H_0^1(\mathcal{U})$.

The following class of parametric elliptic partial differential equations is considered in several papers [18, 20, 21],

$$-\nabla \cdot (a(\cdot, \boldsymbol{x})\nabla u) = b,\ u|_{\partial\mathcal{U}} = 0,$$

where $a(\cdot, \boldsymbol{x})$ is a diffusion coefficient. Considering these parameters as variables $\boldsymbol{x}$ as well as physical variables $\boldsymbol{t}$, by the Karhunen-Loève expansion[18], a stochastic process, $a(\boldsymbol{t}, \boldsymbol{x})$, can be represented as the expectation, $\bar{a}(\boldsymbol{t})$, plus an infinite sum of eigenfunctions of the covariance function multiplied by eigenvalues. We look at a simple case where $a$ affinely depends on $\boldsymbol{x}$ as follows,

$$a(\boldsymbol{t}, \boldsymbol{x}) = \bar{a}(\boldsymbol{t}) + \sum_{j\geq 1} x_j \psi_j(\boldsymbol{t}).$$

Accordingly, $u$ can be represented by a (possibly) infinite expansion in $T_{\boldsymbol{n}}$, typically, tensorized Chebyshev or Legendre functions,

$$u(\boldsymbol{t}, \boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathcal{F}} d_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x}),$$

where $d_{\boldsymbol{n}} \in H_0^1(\mathcal{U})$. When $G$ is a linear functional,

$$g(\boldsymbol{x}) = G(u(\boldsymbol{x})) = \sum_{\boldsymbol{n}\in\mathcal{F}} G(d_{\boldsymbol{n}}) T_{\boldsymbol{n}}(\boldsymbol{x}),$$

and if $c_{\boldsymbol{n}} := G(d_{\boldsymbol{n}})$ and we truncate this expansion to a finite sum of a few $c_{\boldsymbol{n}}$ large in their magnitude,

$$g(\boldsymbol{x}) \approx \sum_{\boldsymbol{n}\in\mathcal{S}\subset\mathcal{F}} c_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x}),$$

which means that a function $g$ can be approximated by a linear combination of a few tensorized basis functions.

Our generalized method is inspired by the sparse approximation of QoI which can be considered as a function of many variables in UQ. We consider a $D$-variate function $f(\boldsymbol{x}) : \mathcal{D} \to \mathbb{C}$ with $\mathcal{D} \subset \mathcal{R}^D$ which is compressible in a bounded orthonormal system $\{T_{\boldsymbol{n}}\}$, i.e.,

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in [M]^D} c_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x})$$

where only $s \ll M^D$ number of $c_{\boldsymbol{n}}$ are significant, such $\boldsymbol{n}$ satisfy $\|\boldsymbol{n}\|_0 \leq d$ with $d \leq D$, and $[M] := \{0, 1, 2, \cdots, M-1\}$. We aim at recovering $s$ significant coefficients $c_{\boldsymbol{n}}$ and corresponding index vectors $\boldsymbol{n}$ efficiently in terms of both runtime and sampling.

Our method is a greedy method motivated by CoSaMP[22], HTP[23], etc. The support identification in these methods requires a construction of a measurement matrix and its multiplication with a sample vector resulting in huge memory and operation counts when considering the high dimensional problems. Accordingly, our method introduces a faster support identifying method that finds the entries of the multi-index $\boldsymbol{n}$ (entry identification) and then integrates the entries to recover the multi-indices (pairing). After estimating the support of $\boldsymbol{n}$, the least squares problem restricted to this support is solved to approximate the corresponding coefficients. The process so far is repeated until we recover the function $f$ with desirable error.

To wrap up, this thesis explores several versions of sublinear-time algorithms for approximating high dimensional functions which show sparsity in some bounded orthonormal product basis(BOPB) expansion. Our algorithm using the CoSaMP approach can compute the discrete Fourier transform quickly instead of our proposed high dimensional sparse Fourier transforms. However, the sparse Fourier transforms shows faster and more efficient performance whereas the sublinear CoSaMP works for the general BOPB including the tensorized

Fourier basis.

In the remainder of the introduction, we review the related work. The first sparse Fourier algorithm was proposed in [4]. The authors introduced a randomized algorithm with $\mathcal{O}(s^2\log^c M)$ runtime and $\mathcal{O}(s^2\log^c M)$ samples where $c$ is a positive number that varies depending on the trade-off between efficiency and accuracy. An algorithm with improved runtime $\mathcal{O}(s\log^c M)$ and samples $\mathcal{O}(s\log^c M)$ was given in [5]. The algorithms given in [6] and [7] achieved $\mathcal{O}(s\log M\log M/s)$ average-case runtime, and the actual empirical results are given in [7] comparing their algorithms with FFTW and an existing sparse FFT from [24]. The algorithms in [4, 5, 6, 7] are all randomized. The first deterministic algorithm using a combinatorial approach was introduced in [8]. In [1], another deterministic algorithm was given whose procedure recognizes frequencies in a similar manner to [6]. The two methods in [1, 6] were published at the same time and both use the idea of working with two sets of samples, one at $\mathcal{O}(s)$ points and the second at the same $\mathcal{O}(s)$ points plus a small shift. The ratio of the FFT of the two sets of points, plus extra machinery, leads to fast deterministic algorithms. The first deterministic algorithm [8] has $\mathcal{O}(s^2\log^4 M)$ runtime and sampling complexity, and in [11], an improved deterministic algorithm was introduced, and the extension to higher dimensional functions was suggested but it suffered the exponential dependence of runtime complexity on the dimension. Another deterministic algorithm was introduced in [1] which uses the similar idea in the frequency recovery through phaseshift and works for noiseless samples from exactly $s$-sparse functions. It has $\mathcal{O}(s\log s)$ runtime and $\mathcal{O}(s)$ sampling complexity on average. In [2], a multiscale method was introduced which works when we are given noisy samples and has $\mathcal{O}(s\log s\log M/s)$ runtime and $\mathcal{O}(s\log M/s)$ sampling complexity.

Extension of one-dimensional sparse Fourier transform to multidimensional problem setting is not straightforward. One simple way of extension is to unwrap the multi-dimensional signal to one-dimensional signal, however, it suffers from the exponentially large runtime complexity due to the curse of dimensionality [11]. The first randomized algorithm for the two-dimensional problem was introduced in [9] using parallel projections of frequencies. In [25], a general $D$-dimensional deterministic sparse Fourier algorithm achieves $\mathcal{O}(Ds^2M)$ samples and $\mathcal{O}(Ds^3 + ds^2M\log(sM))$ runtime complexity by using rank-1 lattices and finding frequencies in an entry-wise fashion. To reduce the runtime complexity, the authors also introduced a randomized version of the algorithm with $\mathcal{O}(Ds + DM)$ samples and $\mathcal{O}(Ds^3)$ runtime. A randomized algorithm introduced in [26] requires $2^{\mathcal{O}(D^2)}(s\log M^D \operatorname{loglog} M^D)$ samples and $2^{\mathcal{O}(D^2)}s\log^{D+3} M^D$ runtime.

Many functions in UQ problems assume sparsity(or compressibility) in various forms that inspires us to consider the possible contribution of compressive sensing and sparse approximation. In [27], randomized algorithms approximating sparse polynomials are introduced. In [25, 28, 29, 30], different versions of high-dimensional sparse Fourier transforms are introduced. As the sparsity in high-dimensional Chebyshev and Lengendre space is considered in some UQ problems, fast algorithms to recover functions with such sparsity have been developed [31, 20, 19]. These papers often have additional assumptions on the structure of the sparsity which implies the degrees of the polynomials with large coefficients are very small. A simple example is the case where a function of larger number of variables actually depends on a fewer variables[32]. Hyperbolic cross[33, 34] and lower sets[19] are also examples with such characteristic, and this implies the bounded mixed smoothness of the objective function instead of arbitrary smoothness. In [3], it is shown that the number of function evaluations required to approximate a function with the arbitrary smoothness cannnot avoid the curse of dimensionality, i.e., super-exponential growth with the number of dimensions. Instead, many methods assumes the structured sparsity in order to avoid the curse of dimensionality in certain problems. One method is a sparse grid method [15] which is used for both interpolation and integration of multi-variate functions. In [20, 18], compressive sensing method with weighted $l_1$ minimization is combined with Petrov-Galerkin method to approximate the linear functional of parametric PDE's solution which can be represented by the combination of a few tensorized Chebyshev polynomials.

## 1.1 Thesis Outline

In the thesis, we present three different approaches to quickly approximating functions of many variables that are sparse in BOPBs using a few function evaluations. A tensorized Fourier basis is the one with good properties compared to the other BOPBs. We develop very efficient sparse FFTs using these advantages. The first approach works for exactly sparse functions by using noiseless samples, and the second approach works for functions with moderate noise. Then, general sparse harmonic transforms are established so that they work for any BOPB including the Fourier basis.

As a first step to our goal of a high dimensional sparse FFT, the thesis addresses the case for the continuous functions without noise in a high dimensional setting in the Chapter 2. We introduce effective methods to address the registration and the collision problems in Section 2.1. In Section 2.2, the ideas of various methods to resolve these problems in 2D are introduced which can be extended to the methods in the higher dimensional setting. In particular, we introduce a novel *partial unwrapping* technique in Section 2.3 that is shown to be highly effective in reducing the registration and collision complexity while maintains the sublinear runtime efficiency even in very high-dimensional problems. We shall show in

Section 2.4 that our algorithm can achieve $\mathcal{O}(Ds\log s)$ computational complexity and $\mathcal{O}(Ds)$ sampling size on average assuming there is no worst-case scenario. In Section 2.5, we present as examples computational results for sparse FFT where the dimensions are 100 and 1000 respectively. For comparison, the traditional $D$-dimensional FFT requires $\mathcal{O}(M^D\log M^D)$ time complexity and $\mathcal{O}(M^D)$ sampling complexity, which is impossible to implement on any computers today.

In Chapter 3, we shall present an adaptation of the algorithm for noisy data. In Section 3.1 we introduce our problem setting, necessary notation and our noise model. Based on these, a multiscale method for frequency entry estimate is introduced and analyzed in Section 3.2. In Section 3.3, parameters that determine the performance of our algorithm are introduced and the pseudocode is given with description. The results of the numerical experiments are shown in Section 3.4.

In Chapter 4, a generalized sparse harmonic transform is described. In Section 4.1, we introduce the notation that is used throughout the chapter and the problem setting, and reviewed the basics of compressive sensing related to the thesis. In Section 4.2 the description and the performance results of our proposed method are introduced. The analysis supporting the results in Section 4.2 is elaborated in Section 4.3 and the results of numerical experiments are shown in Section 4.4.

# Chapter 2

# High-dimensional Sparse Fourier Transforms

## 2.1 Preliminaries

### 2.1.1 Review of the One-Dimensional Sparse Fourier Transform

The one-dimensional sublinear sparse Fourier algorithm inspiring our method was developed in [1]. We briefly introduce the idea and notation of the algorithm before developing the multidimensional ones throughout this chapter. We assume a function $f : [0, 1) \to \mathbb{C}$ with sparsity $s$ as the following,

$$f(x) \;=\; \sum_{n \in \mathcal{S}} c_n e^{2\pi i n x} \tag{2.1}$$

where the bandwidth is $M$, i.e., each frequency $n \in \mathcal{S} \subset [-M/2, M/2) \cap \mathbb{Z}$, the cardinality, $|\mathcal{S}|$, of $\mathcal{S}$ is $s$, and the corresponding nonzero coefficient $c_n$ is in $\mathbb{C}$ for all $n$. We can consider it as a periodic function over $\mathbb{R}$ instead of $[0, 1)$. The goal of the algorithm is to recover all coefficients $c_n$ and frequencies $n$ so that we can reconstruct the function $f$. This algorithm is called the "phase-shift" method since it uses equi-spaced samples from the function and those at the positions shifted by a small positive number $\epsilon$. To verify that the algorithm correctly finds the frequencies in the bandwidth $M$, $\epsilon$ should be strictly no bigger than $1/M$.

We denote a sequence of samples shifted by $\epsilon$ with the sampling rate $1/p$, where $p$ is a prime number, as

$$\boldsymbol{f}_{p,\epsilon} = \Big(f(0+\epsilon), f(\frac{1}{p}+\epsilon), f(\frac{2}{p}+\epsilon), f(\frac{3}{p}+\epsilon), \cdots, f(\frac{p-1}{p}+\epsilon)\Big). \tag{2.2}$$

We skip much of the details here. In a nutshell, choosing $p$ slightly larger than $s$ is enough to make the algorithm work. In [1] $p$ is set to be roughly $5s$, which is much smaller than the Nyquist rate $M$. Discrete Fourier transform (DFT) is then applied to the sample sequence $\boldsymbol{f}_{p,\epsilon}$, and the $h^{\text{th}}$ element of its result is the following

$$\mathcal{F}(\boldsymbol{f}_{p,\epsilon})[h] = p \sum_{n=h(\bmod p)} c_n e^{2\pi i \epsilon n} \tag{2.3}$$

where $h = 0, 1, 2, \ldots, p-1$. If there is only one frequency $n$ congruent to $h$ modulo $p$,

$$\mathcal{F}(\boldsymbol{f}_{p,\epsilon})[h] = p c_n e^{2\pi i \epsilon n}. \tag{2.4}$$

By putting 0 instead of $\epsilon$, we can get the unshifted samples $\boldsymbol{f}_{p,0}$ and applying the DFT gives

$$\mathcal{F}(\boldsymbol{f}_{p,0})[h] = p c_n. \tag{2.5}$$

This process so far is visualized in the Figure 2.1. As long as there is no collision of frequencies with modulo $p$, we can find frequencies and their corresponding coefficients by the following computation

$$n = \frac{1}{2\pi\epsilon}\text{Arg}\Big(\frac{\mathcal{F}(\boldsymbol{f}_{p,\epsilon})[h]}{\mathcal{F}(\boldsymbol{f}_{p,0})[h]}\Big),$$

$$c_n = \frac{1}{p}\mathcal{F}(\boldsymbol{f}_{p,0})[h], \tag{2.6}$$

where the function "Arg" gives us the argument falling into $[-\pi, \pi)$. Note that $n$ should be the only frequency congruent to $h$ modulo $p$, i.e., $n$ has no collision with other frequencies modulo $p$. The test to determine whether a collision occurs or not is

$$\frac{|\mathcal{F}(\boldsymbol{f}_{p,\epsilon})[h]|}{|\mathcal{F}(\boldsymbol{f}_{p,0})[h]|} = 1. \tag{2.7}$$

The equality (2.7) above holds when there is no collision. If there is a collision, the equality does not hold for almost all $\epsilon$, i.e., the test fails to predict a collision for the finite number of $\epsilon$ [1]. Furthermore, it is also shown in [1] that for any $\epsilon = \frac{a}{b}$ with $a, b$ coprime and $b \geq 2M$, equality (2.7) does not hold for at least one shift in a certain finite set of integer multiples of $\epsilon$ unless there is no collision. In practical implementations, we choose $\epsilon$ to be $1/CM$ for some positive integer $C \geq 1$ and allow some small difference $\tau$ between the left and right sides of (2.7) where $\tau$ is a very small positive number.

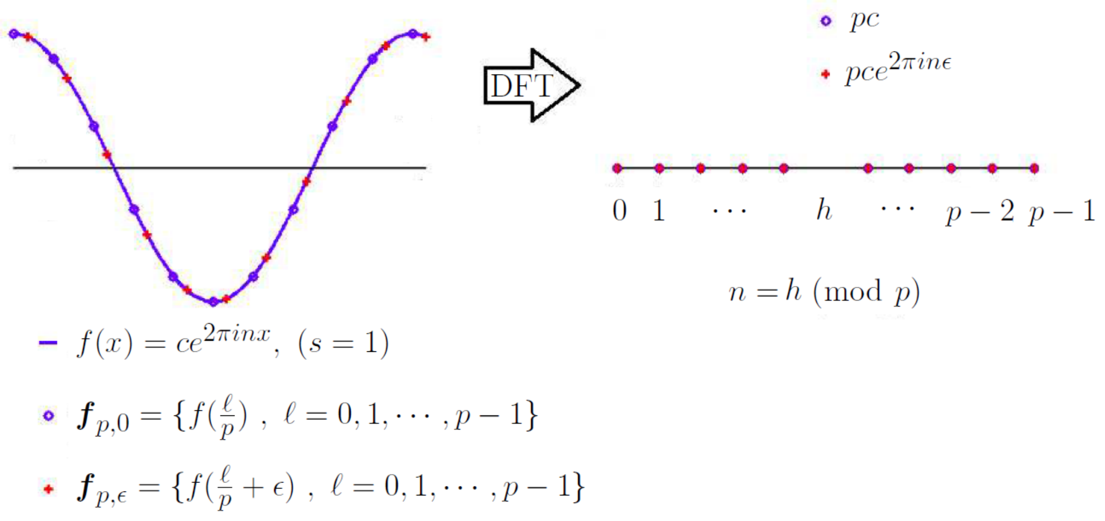

DFT
$pc$
$pce^{2\pi i n\epsilon}$
$0$ $1$ $\cdots$ $h$ $\cdots$ $p-2$ $p-1$
$n = h \pmod{p}$
$f(x) = ce^{2\pi i n x}$, $(s = 1)$
$\boldsymbol{f}_{p,0} = \{f(\frac{\ell}{p})\ ,\ \ell = 0, 1, \cdots, p-1\}$
$\boldsymbol{f}_{p,\epsilon} = \{f(\frac{\ell}{p} + \epsilon)\ ,\ \ell = 0, 1, \cdots, p-1\}$

Figure 2.1: Process of 1D sublinear sparse Fourier algorithm

The above process is one loop of the algorithm with a prime number $p$. To explain it from a different view point, we can imagine that there are $p$ bins. Then we sort all frequencies into these bins according to their remainder modulo $p$. If there are more than one frequencies in one bin, then a collision happened. If there is only one frequency, then there is no collision. To determine whether a collision occurs, we use the above test. In the case where the test fails, i.e., the ratio is not 1, we need to use another prime number $p'$. Thus we re-sort the frequencies into $p'$ bins by their remainder modulo $p'$. Even if two frequencies collide modulo $p$, it is likely that they do not collide modulo $p'$. Particularly, the Chinese Remainder Theorem guarantees that with a finite set of prime numbers, $\{p_\ell\}$, any frequency within the bandwidth $M$ can be uniquely identified, given $\prod_\ell p_\ell \geq M$. Algorithmically, for each loop, we choose a different prime number $p'$ and repeat equations (2.2)–(2.7) with $p$ replaced by $p'$. In this way we can recover all $c_n$ and $n$ in sublinear time $\mathcal{O}(s\log s)$ using $\mathcal{O}(s)$ samples on average. The overall code is shown in Algorithm 4 referred from [1].

### 2.1.2 Multidimensional Problem Setting and Worst Case Scenario

In this section, the multidimensional problem is introduced. Let us consider a function $f : \mathbb{R}^D \to \mathbb{C}$ such that

$$f(\boldsymbol{x}) \;=\; \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{x}}, \tag{2.8}$$

where $\boldsymbol{n} \in [-M/2, M/2)^D \cap \mathbb{Z}^D$ and $c_{\boldsymbol{n}} \in \mathbb{C}$. That is, from (2.1), $x$ is replaced by the $D$-dimensional phase or time vector $\boldsymbol{x}$, the frequency $n$ is replaced by the frequency vector $\boldsymbol{n}$, and thus the operator between $\boldsymbol{n}$ and $\boldsymbol{x}$ is a dot product instead of simple scalar multiplication. We can see that this is a natural extension of the one-dimensional sparse problem. As in the one-dimensional setting, if we find $c_{\boldsymbol{n}}$ and $\boldsymbol{n}$, we recover the function $f$.

**Algorithm 1** Phaseshift

```
1: procedure Phaseshift
2: Input: $f, C, s, M, \epsilon$
3: Output: $R$
4:     $R \leftarrow \emptyset$
5:     $t \leftarrow 1$
6:     while $|R| < s$ do
7:         $s^* \leftarrow s - |R|$
8:         $p \leftarrow t^{\text{th}}$ prime number $\geq Cs^*$
9:         $Q(x) = \sum_{(n,c_n)\in R} c_n e^{2\pi i n x}$
10:        for $\ell = 0 \to p-1$ do
11:            $f_{p,0}[\ell] = f(\frac{\ell}{p}) - Q(\frac{\ell}{p})$
12:            $f_{p,\epsilon}[\ell] = f(\frac{\ell}{p} + \epsilon) - Q(\frac{\ell}{p} + \epsilon)$
13:        end for
14:        $\mathcal{F}(\boldsymbol{f}_{p,\epsilon}) = \text{FFT}(\boldsymbol{f}_{p,\epsilon})$
15:        $\mathcal{F}(\boldsymbol{f}_{p,0}) = \text{FFT}(\boldsymbol{f}_{p,0})$
16:        $\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,0}) = \text{SORT}(\mathcal{F}(\boldsymbol{f}_{p,0}))$
17:        for $h = 0 \to s^* - 1$ do
18:            if $\left|\frac{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,0})[h]|}{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,\epsilon})[h]|} - 1\right| < \epsilon$ then
19:                $\widetilde{n} = \frac{1}{2\pi\epsilon}\text{Arg}\left(\frac{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{\text{p},\epsilon})[\text{h}]}{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{\text{p},0})[\text{h}]}\right)$
20:                $c_{\widetilde{n}} = \frac{1}{p}\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,0})[h]$
21:                $R \leftarrow R \cup (\widetilde{n}, c_{\widetilde{n}})$
22:            end if
23:        end for
24:        prune small coefficients from $R$
25:        $t \leftarrow t + 1$
26:    end while
27: end procedure
```

However, since our time and frequency domain have changed, we cannot apply the previous algorithm directly. If we project the frequencies onto a line, then we can apply the former algorithm so that we can retain the sublinear time complexity. Since the operator between frequency and time vectors is a dot product, we can convert projection of frequencies to that of time. For example, we consider the projection onto the first axis, that is, we put the last $D-1$ elements of time vectors as 0. If the projection is one-to-one, i.e., there is no collision, then we can apply the algorithm in Section 2.1.1 to this projected function to recover the first element of frequency vectors. If there is a collision on the first axis, then we can try another projection onto $\widetilde{k}$th axis, $\widetilde{k} = 2, 3, \cdots, D$, until there are no collisions. We introduce in the latter sections how to recover the corresponding remaining $D - 1$ elements by extending the test to determine the occurrence of a collision in Section 2.1.1. Furthermore, to reduce the chance of a collision through projections, we use an "unwrapping method" which unwraps frequencies onto a lower dimension guaranteeing a one-to-one projection. There are both a "full unwrapping" and a "partial unwrapping" methods, which are explained in later sections.

We shall call projections onto any one of the coordinate axes a *parallel projection.* The worst case is where there is a collision for every parallel projection. This obviously happens when a subset of frequency vectors form the vertices of a $D$-dimensional hypercube, but it can happen also with other configurations that require fewer vertices. Then our method cannot recover any of these frequency vectors via parallel projections. To resolve this problem, we introduce *tilted projections*: instead of simple projection onto axes we project frequency vectors onto tilted lines or planes so that there is no collision after the projection. We shall call this the *tilting method* and provide the details in the next section. After introducing these projection methods, we explore which combination of these methods is likely to be optimal.

## 2.2 Two Dimensional Sublinear Sparse Fourier Algorithm

As means of explanation, we introduce the two-dimensional case in this section and extend this to higher dimensions in Section 2.3. The basic two-dimensional algorithm using a parallel projection is introduced in Section 2.2.1, the full unwrapping method is introduced in Section

2.2.2 and the tilting method for the worst case is discussed in Section 2.2.3.

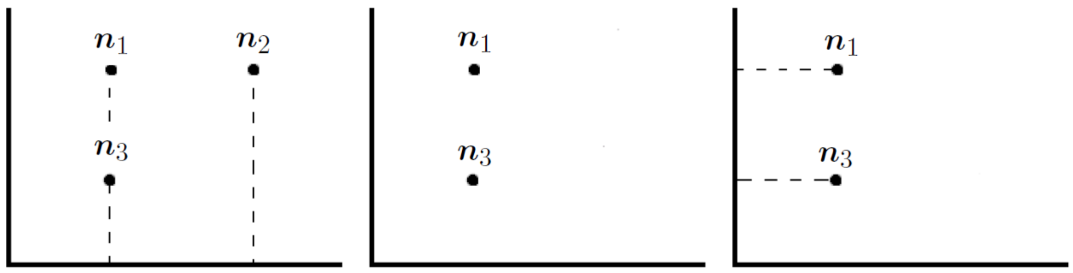


Figure 2.2: Process of the basic algorithm in 2D

### 2.2.1 Basic Algorithm Using Parallel Projection

Our basic two-dimensional sublinear algorithm excludes certain worst case scenarios. In most cases, we can recover frequencies in the 2-D plane by projecting them onto each horizontal axis or vertical axis. Figure 2.2 is a simple illustration. Here we have three frequency vectors where $\boldsymbol{n}_1$ and $\boldsymbol{n}_3$ are colliding with each other when they are projected onto the horizontal axis, and $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are colliding when they are projected onto the vertical axis. The first step is to project the frequency vectors onto the horizontal axis and recover $\boldsymbol{n}_2$ and its corresponding coefficient $c_{\boldsymbol{n}_2}$ only, since it is not colliding. After subtracting $\boldsymbol{n}_2$ from the data, we project the remaining frequency vectors onto the vertical axis and then find both $\boldsymbol{n}_1$ and $\boldsymbol{n}_3$.

Now let us consider the generalized two-dimensional basic algorithm. Assume that we have a two-dimensional function $f$ with sparsity $s$ :

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{x}}, \quad c_{\boldsymbol{n}} \in \mathbb{C}, \quad \boldsymbol{n} \in \Big[-\frac{M}{2}, \frac{M}{2}\Big)^2 \cap \mathbb{Z}^2. \tag{2.9}$$

For now, let us focus on one frequency vector $\widetilde{\boldsymbol{n}}$ which is not collided with any other pairs when they are projected onto the horizontal axis. To clarify put $\boldsymbol{x} = (x_1, 0)$ with $\boldsymbol{n} = (n_1, n_2)$ into (2.9),

$$f^1(x_1) \;:=\; f(x_1, 0) \;=\; \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i n_1 x_1}, \tag{2.10}$$

which gives the same effect of the parallel projection of frequency vectors. Now, we can consider this function as a one-dimensional function $f^1$ so that we can use the original one dimensional sparse Fourier algorithm to find the first component of $\widetilde{\boldsymbol{n}}$. We get the samples $\boldsymbol{f}_p^1$ and $\boldsymbol{f}_{p,\epsilon}^{1,1}$ without and with the shift by $\epsilon$, respectively. We can find these in the form of sequences in (2.2), apply the DFT to them, and then recover the first component of the frequency pair and its coefficient as follows,

$$\begin{aligned} \widetilde{n}_1 &= \frac{1}{2\pi\epsilon}\mathrm{Arg}\Big(\frac{\mathcal{F}(\boldsymbol{f}_{\mathrm{p},\epsilon}^{1,1})[\mathrm{h}]}{\mathcal{F}(\boldsymbol{f}_{\mathrm{p}}^{1})[\mathrm{h}]}\Big), \\ c_{\widetilde{\boldsymbol{n}}} &= \frac{1}{p}\mathcal{F}(\boldsymbol{f}_p^1)[h]. \end{aligned} \tag{2.11}$$

At the same time, we need to find the second component. In (2.10), we replace 0 by $\epsilon$. Then

$$\begin{aligned} f^{1,2}(x_1) \;:=\; f(x_1, \epsilon) &= \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i(n_1 x_1 + n_2\epsilon)}, \\ \mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{1,2})[h] &= p c_{\widetilde{\boldsymbol{n}}} e^{2\pi i \widetilde{n}_2 \epsilon}, \\ \widetilde{n}_2 &= \frac{1}{2\pi\epsilon}\mathrm{Arg}\Big(\frac{\mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{1,2})[h]}{\mathcal{F}(\boldsymbol{f}_p^1)[h]}\Big), \end{aligned} \tag{2.12}$$

where $\boldsymbol{f}_{p,\epsilon}^{1,2}$ are samples shifted by $\epsilon$ in the vertical sense with rate $1/p$ from the function $f^{1,2}$. The equalities in (2.11) hold only when $\widetilde{n}_1$ is the only one congruent to $h$ modulo $p$ among every first component of $s$ frequency pairs and (2.12) holds only when the previous condition is satisfied and $\widetilde{\boldsymbol{n}} = (\widetilde{n}_1, \widetilde{n}_2)$ does not collide with other frequency pairs from the parallel projection.

Now we have two kinds of collisions. The first one is from taking modulo $p$ after the parallel projection and the second one is from the projection. Thus we need two tests. To determine whether there are both kinds of collisions, we use similar tests as (2.7). If there are at least two different $n_1$ congruent to $h$ modulo $p$, then the second equality in the following is not satisfied for almost all $\epsilon$, just as (2.7),

$$\frac{|\mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{1,1})[h]|}{|\mathcal{F}(\boldsymbol{f}_p^1)[h]|} = \frac{|p\sum_{n_1=h(\bmod p)} c_{\boldsymbol{n}} e^{2\pi i\epsilon n_1}|}{|p\sum_{n_1=h(\bmod p)} c_{\boldsymbol{n}}|} = 1. \tag{2.13}$$

Likewise, if there is a collision from the projection, i.e., the first components $n_1$'s of at least two frequency vectors are identical and the corresponding $n_2$'s are different, the following second equality does not hold for almost all $\epsilon$,

$$\frac{|\mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{1,2})[h]|}{|\mathcal{F}(\boldsymbol{f}_p^1)[h]|} = \frac{|p\sum_{n_1=h(\bmod p)} c_{\boldsymbol{n}} e^{2\pi i\epsilon n_2}|}{|p\sum_{n_1=h(\bmod p)} c_{\boldsymbol{n}}|} = 1. \tag{2.14}$$

The two tests above are both satisfied only when there is no collision both from taking modulo $p$ and the projection. We use these for the complete recovery of the objective frequencies.

So far we project the frequencies onto the horizontal axis. After we find the non-collided frequencies from the first projection, we subtract a function consisting of the found frequencies and their coefficients from the original function $f$ to get a new function. Next we project this new function onto the vertical axis and do a similar process. The difference is to exchange 1 and 2 in the super-indices in (2.10) through (2.14). Again, find the remaining non-collided frequencies, change the axis again and keep doing this until we recover all of the frequencies.

### 2.2.2 Full Unwrapping Method

We introduce another kind of projection which is one-to-one. The full unwrapping method uses one-to-one projections onto one-dimensional lines instead of the parallel projection onto axes from the previous method. We consider the $s$ pairs of frequencies $\boldsymbol{n} = (n_1, n_2) \in \mathcal{S}$ and transform them as follows

$$(n_1, n_2) \;\to\; n_1 + Mn_2. \tag{2.15}$$

This transformation in frequency space can be considered as the transformation in phase or time space. That is, from the function in (2.9)

$$g(x) \;:=\; f(x, Mx) \;=\; \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i (n_1 + Mn_2)x}. \tag{2.16}$$

The function $g(x)$ is a one-dimensional function with the sparsity $s$ and the bandwidth bounded by $M^2$. We can apply the algorithm in Section 2.1.1 on $g$ so that we recover $s$ frequencies of the form on the right side of the arrow in (2.15). Whether unwrapped or not, the coefficients are the same, so we can find them easily. In the end we need to wrap the unwrapped frequencies to get the original pairs. Remember that unwrapping transformation is one-to-one. Thus we can wrap them without any collision.

Since the pairs of the frequencies are projected onto the one-dimensional line directly, we call this method the "full unwrapping method". A problem of this method occurs when the dimension $D$ gets large. From the above description, we see that after the one-to-one unwrapping the total bandwidth of the two dimensional signal increases from $M$ in each dimension to $M^2$. If the full unwrapping method is applied to a function in $D$-dimensions, then to guarantee the one-to-one transformation the bandwidth will be $M^D$. Theoretically this does not matter. However, since $\epsilon$ is dependent on the bandwidth, in the case where $D$ is large, we need to consider the limit of machine precision for practical implementations. As a result, we need to introduce the partial unwrapping method to prevent the bandwidth from becoming too large. The partial unwrapping method is discussed in Section 2.3.

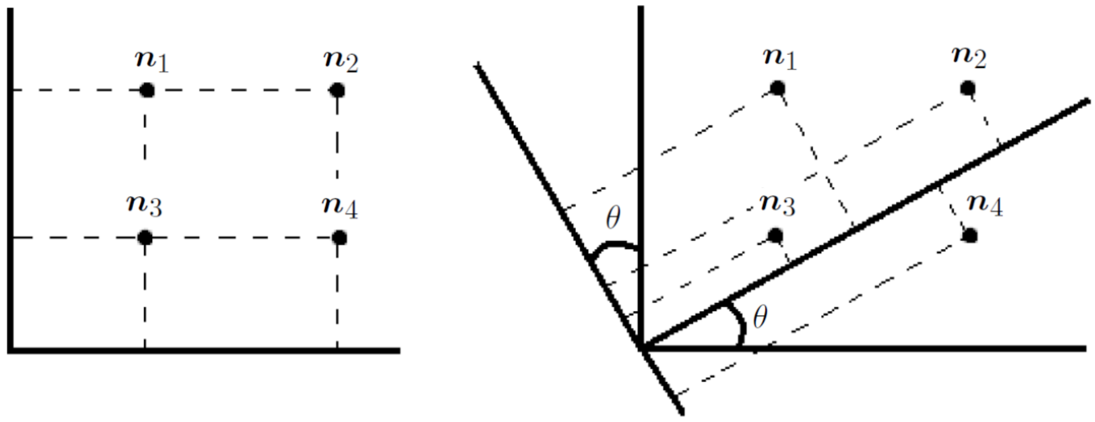


Figure 2.3: Worst case scenario in two dimensions and solving it through tilting

### 2.2.3 Tilting Method for the Worst Case

Up till now, we have assumed that we do not encounter the worst case, i.e., that we do not encounter the case where any frequency pair has collisions from the parallel projection for all coordinate axes. This makes the algorithm break down. A very simple example of the worse case scenario is given in Figure 2.3 where the four frequency pairs $\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3$ and $\boldsymbol{n}_4$ form a rectangle, and thus each pair has collisions in both horizontal and vertical projections. The following method is for finding those frequency pairs. Basically, we rotate axes of the frequency plane and thus use a projection onto a one-dimension system which is a tilted line with the tilt chosen so that there are no collisions. If the horizontal and vertical axes are rotated with angle $\theta$ then the frequency pair $\boldsymbol{n} = (n_1, n_2)$ can be relabeled with new coordinates as the right side of the following

$$(n_1, n_2) \;\rightarrow\; (n_1 \cos\theta - n_2 \sin\theta, n_1 \sin\theta + n_2 \cos\theta). \tag{2.17}$$

In phase-sense, this rotation can be written as

$$\begin{aligned} g(\tilde{x}_1, \tilde{x}_2) &:= f(\tilde{x}_1 \cos\theta + \tilde{x}_2 \sin\theta, -\tilde{x}_1 \sin\theta + \tilde{x}_2 \cos\theta) \\ &= \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \{n_1(\tilde{x}_1 \cos\theta + \tilde{x}_2 \sin\theta) + n_2(-\tilde{x}_1 \sin\theta + \tilde{x}_2 \cos\theta)\}}, \\ &= \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \{(n_1 \cos\theta - n_2 \sin\theta)\tilde{x}_1 + (n_1 \sin\theta + n_2 \cos\theta)\tilde{x}_2\}}. \end{aligned} \tag{2.18}$$

We can apply the basic algorithm in Section 2.2.1 to the function $g$ to get the frequency pairs in the form of the right side of the arrow in (2.17).

One problem we face is that the components of the projected frequency pairs should be integers to apply the method, since we assume the integer frequencies in the first place. To guarantee the injectivity for both projections, $\tan\theta$ should be irrational, however, and thus the projected frequencies become irrational. Thus, we should try a rational $\tan\theta$, and to make them integer it is inevitable to increase the bandwidth by multiplying the least common multiple of the denominators of $\sin\theta$ and $\cos\theta$. We assume the following with integers $a$, $b$ and $c$

$$\sin\theta \;=\; \frac{a}{c}, \quad \cos\theta \;=\; \frac{b}{c}, \quad \gcd(a, c) \;=\; \gcd(b, c) \;=\; 1. \tag{2.19}$$

Multiplying $c$ to both inputs in the right-hand side of (2.18) we obtain

$$\begin{aligned}\hat{g}(\tilde{x}_1, \tilde{x}_2) &:= f(c(\tilde{x}_1 \cos\theta + \tilde{x}_2 \sin\theta), c(-\tilde{x}_1 \sin\theta + \tilde{x}_2 \cos\theta)) \\ &= \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \{(cn_1 \cos\theta - cn_2 \sin\theta)\tilde{x}_1 + (cn_1 \sin\theta + cn_2 \cos\theta)\tilde{x}_2\}}. \qquad (2.20)\end{aligned}$$

As long as there is no collision for at least one projection, the frequency pairs, $(cn_1 \cos\theta - cn_2 \sin\theta, cn_1 \sin\theta + cn_2 \cos\theta)$, can be found by applying the basic algorithm in Section 2.2.2 on $\hat{g}$. Due to the machine precision the integer $c$ should not be too large, or the bandwidth gets too large resulting in the failure of the algorithm. If four pairs of frequencies are at vertices of a rectangle aligned with coordinate axes before the rotation, then they are not aligned after the rotation with $0 < \theta < \pi/2$. Thus we can assure finding whole frequencies whether they are in the worst case or not.

The pseudo code of the 2D tilting method is shown in Algorithm 2. The lines 15 and 16 mean that each frequency pair $(n_1, n_2)$ is rotated by a matrix $[\cos\ -\sin;\ \sin\ \cos]$ and scaled to make the rotated components integers. Thus we first find the frequency pairs in the form of $\widetilde{\boldsymbol{n}} = (n_1 \cos - n_2 \sin, n_1 \sin + n_2 \cos)$ and after finding all of them, we rotate them back into the original pairs with the matrix $[\cos\ \sin;\ -\sin\ \cos]$ in line 39.

This tilting method is a straight forward way to resolve the worst case problem. First, we recover the frequencies as much as possible from the basic parallel projection method. If we cannot get any frequency pairs for several projections switching among each axis then, assuming that the worst case happens, we apply the tilting method with an angle so that all remaining frequency pairs are found. We only introduced the tilting method in the two-dimensional case, but the idea of rotating the axes can be extended to the general $D$-dimensional case with some effort. On the other hand, we may notice that the probability of

**Algorithm 2** 2D Sparse Fourier Algorithm with Tilting Method Pseudo Code

```
 1: procedure 2DTiltedPhaseshift
 2: Input: $f, C, s, M, D, \epsilon$, base, height, hypo
 3: Output: $R$
 4:     $R \leftarrow \emptyset$
 5:     $t \leftarrow 1$
 6:     cos ← base, sin ← height
 7:     while $|R| < s$ do
 8:         $s^* \leftarrow s - |R|$
 9:         $p \leftarrow t^{\text{th}}$ prime number $\geq Cs^*$
10:         $\widetilde{k} \leftarrow (t \bmod 2) + 1$
11:         $Q(\widetilde{\boldsymbol{x}}) = \sum_{(\widetilde{\boldsymbol{n}}, c_{\widetilde{\boldsymbol{n}}}) \in R} c_{\widetilde{\boldsymbol{n}}} e^{2\pi i \widetilde{\boldsymbol{n}} \cdot \widetilde{\boldsymbol{x}}}$
12:         for $k = 1 \to 2$ do
13:             for $\ell = 0 \to p - 1$ do
14:                 $m' \leftarrow \widetilde{k} \bmod 2$, $m'' \leftarrow \widetilde{k} + 1 \bmod 2$, $n' \leftarrow k \bmod 2$, $n'' \leftarrow k + 1 \bmod 2$
15:                 $f_{p,\epsilon}^{\widetilde{k},k}[\ell] = f((\frac{\ell}{p}m' + \epsilon n')\cos + (\frac{\ell}{p}m'' + \epsilon n'')\sin, -(\frac{\ell}{p}m' + \epsilon n')\sin + (\frac{\ell}{p}m'' + \epsilon n'')\cos)$
                            $-Q(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon \boldsymbol{e}_k)$
16:                 $f_p^{\widetilde{k},k}[\ell] = f(\frac{\ell}{p}m'\cos + \frac{\ell}{p}m''\sin, -\frac{\ell}{p}m'\sin + \frac{\ell}{p}m''\cos) - Q(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}})$
17:             end for
18:             $\mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k}) = \text{FFT}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k})$
19:             $\mathcal{F}(\boldsymbol{f}_p^{\widetilde{k}}) = \text{FFT}(\boldsymbol{f}_p^{\widetilde{k}})$
20:             $\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}}) = \text{SORT}(\mathcal{F}(\boldsymbol{f}_p^{\widetilde{k}}))$
21:         end for
22:         for $h = 0 \to s^* - 1$ do
23:           $\ell \leftarrow 0$
24:           for $k = 1 \to 2$ do
25:             if $\left| \frac{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k},k})[h]|}{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k})[h]|} - 1 \right| < \epsilon$ then
26:               $\ell \leftarrow \ell + 1$
27:             end if
28:             $\widetilde{\boldsymbol{n}}_k = \frac{1}{2\pi\epsilon}\text{Arg}\left(\frac{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k})[h]}{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}})[h]}\right)$
29:             $c_{\widetilde{\boldsymbol{n}}} = \frac{1}{p}\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}})[h]$
30:           end for
31:           if $\ell == 2$ then
32:             $R \leftarrow R \cup (c_{\widetilde{\boldsymbol{n}}}, \widetilde{\boldsymbol{n}})$
33:           end if
34:         end for
35:         prune small coefficients from $R$
36:         $t \leftarrow t + 1$
37:     end while
38:     cos ← $\frac{\text{base}}{\text{hypo}}$, sin ← $\frac{\text{height}}{\text{hypo}}$
39:     rotate each $\widetilde{\boldsymbol{n}}$ back to $\boldsymbol{n}$ using a matrix [cos sin;− sin cos] and restore it in $R$
40: end procedure
```

this worst case is very low, especially when the number of dimensions $D$ is large. Its details are shown in Section 2.3.

## 2.3 Partial Unwrapping Method for High Dimensional Algorithm

In this section we present the *partial unwrapping* method for a sublinear sparse Fourier algorithm for very high dimensional data. As we have already mentioned, while full unwrapping converts a multi-dimensional problem into a single dimensional problem, it is severely limited in its viability when the dimension is large or when the bandwidth is already high because of the increased bandwidth. Partial unwrapping is introduced here to overcome this problem and other problems. In Section 2.3.1 we give a four dimensional version of the algorithm using the partial unwrapping method as well as a generalize it to $D$ dimension. In Section 2.3.2, the probability of the worst case in $D$ dimension is analyzed.

### 2.3.1 Partial Unwrapping Method

To see the benefit of partial unwrapping we need to examine the main difficulties we may encounter in developing the sublinear sparse Fourier algorithms. For this let us consider a hypothetical case of sparse FFT where we have $s = 100$ frequencies in a 20-dimensional Fourier series distributed in $[-10, 10)^{20}$. When we perform the parallel projection method, because the bandwidth is small, there will be a lot of collisions after the projections. It is often impossible to separate any frequency after each projection, and the task could thus not be completed. This, ironically, is a *curse of small bandwidth* for sparse Fourier algorithm.

On the other hand, if we do the full unwrapping we would have increased the bandwidth to $M' = 20^{20}$, which is impossible to do within reasonable accuracy because $M'$ is too large.

However, a partial unwrapping would reap the benefit of both worlds. Let us now break down the 20 dimensions into 5 lower 4-dimensional subspaces, namely we write

$$[-10, 10)^{20} = \Big([-10, 10)^4\Big)^5.$$

In each subspace we perform the full unwrapping, which yields bandwidth $M' = 20^4 = 160,000$ in the subspace. This bandwidth $M'$ is large enough compared with $s$, so when the projection method is used there is a very good probability that the collision will occur only for a small percentage of the frequencies, allowing them to be reconstructed. On the other hand, $M'$ is not so large that the phase-shift method will incur significant error.

One of the greatest advantage of partial unwrapping is to turn the curse of dimensionality into the *blessing* of dimensionality.

Note that in the above example, the 4 dimensions of any subspace do not have to follow the natural order. By randomizing (if necessary) the order of the dimensions it may achieve the same goal as the tilting method. Also note that the dimension for each subspace needs not be uniform. For example, we can break down the above 20-dimensional example into four 3-dimensional subspaces and two 4-dimensional subspaces, i.e.

$$[-10, 10)^{20} = \Big([-10, 10)^3\Big)^4 \times \Big([-10, 10)^4\Big)^2.$$

This will lead to further flexibility.

#### 2.3.1.1 Example of 4-D Case

Before introducing the generalized partial unwrapping algorithm for dimension $D$, let us think about the simple case of 4 dimensions. We assume that $s$ frequency vectors are in 4-dimensional space ($D = 4$). Then, a function $f$ constructed from these frequency vectors is as follows,

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{x}}, \; c_{\boldsymbol{n}} \in \mathbb{C}, \; \boldsymbol{n} \in \Big[-\frac{M}{2}, \frac{M}{2}\Big)^4 \cap \mathbb{Z}^4. \tag{2.21}$$

Since $4 = 2 \times 2$, the frequency pairs of the two-two dimensional spaces are both unwrapped onto one-dimensional spaces. Here, 4 dimensions is projected onto 2 dimensions as follows

$$\begin{aligned} g(x_1, x_2) &:= f(x_1, Mx_1, x_2, Mx_2) \\ &= \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i\{(n_1+Mn_2)x_1+(n_3+Mn_4)x_2\}} \\ &= \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i(\widetilde{n}_1 x_1+\widetilde{n}_2 x_2)}, \end{aligned} \tag{2.22}$$

where $\widetilde{n}_1 = n_1 + Mn_2$ and $\widetilde{n}_2 = n_3 + Mn_4$. Note that this projection is one-to-one so as to guarantee the inverse transformation.

Now we can apply the basic projection method in Section 2.2.1 to this function $g$ redefined as the 2-dimensional one. To make this algorithm work, $\widetilde{\boldsymbol{n}} = (\widetilde{n}_1, \widetilde{n}_2)$ should not collide with any other frequency pair after the projection onto either the horizontal or vertical axes. If not, we can consider using the tilting method. After finding all the frequencies in the form of $(\widetilde{n}_1, \widetilde{n}_2)$, it can be transformed to $(n_1, n_2, n_3, n_4)$.

### 2.3.1.2 Generalization

We introduce the final version of the multidimensional algorithm in this section. Its pseudo code and detailed explanation are given in Algorithm 3 and Section 2.5.1, respectively. We start with a $D$-dimensional function $f$,

$$f(\boldsymbol{x}) \;=\; \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{x}}, \quad c_{\boldsymbol{n}} \in \mathbb{C}, \quad \boldsymbol{n} \in \Big[-\frac{M}{2}, \frac{M}{2}\Big)^D \cap \mathbb{Z}^D. \tag{2.23}$$

Let us assume that $D$ can be divided into $d_1$ and $d_2$ - the case of $D$ being a prime number will be mentioned at the end of this section. The domain of frequencies can be considered as $([-M/2, M/2) \cap \mathbb{Z})^D = (([-M/2, M/2) \cap \mathbb{Z})^{d_1})^{d_2}$ and $([-M/2, M/2) \cap \mathbb{Z})^{d_1}$ will be reduced to one dimension, as $d_1$ is in the 4 dimensional case. Each of the $d_1$ elements of a frequency vector, $\boldsymbol{n} = (n_1, n_2, \cdots, n_D)$, is unwrapped as

$$\begin{aligned}
&(n_{(d_1q+1)}, n_{(d_1q+2)}, n_{(d_1q+3)}, \cdots, n_{(d_1q+d_1)}) \\
&\;\to\; n_{(d_1q+1)} + Mn_{(d_1q+2)} + M^2 n_{(d_1q+3)} + \cdots + M^{d_1-1} n_{(d_1q+d_1)} \\
&\;=:\; \widetilde{n}_{(q+1)}
\end{aligned} \tag{2.24}$$

with $q = 0, 1, 2, \cdots, d_2 - 1$, increasing the respective bandwidth from $M$ to $M^{d_1}$ and having injectivity. We rewrite this transformation in terms of the phase. With $\boldsymbol{x} = (x_1, x_2, \cdots, x_D)$ and put the following into $x_\ell$

$$M^{R(\ell-1,d_1)} \tilde{x}_{Q(\ell-1,d_1)+1} \tag{2.25}$$

for all $\ell = 1, 2, \cdots, D$, where $R(\ell-1, d_1)$ and $Q(\ell-1, d_1)$ are the remainder from dividing $\ell-1$ by $d_1$ and quotient from dividing $\ell-1$ by $d_1$ respectively, and $\widetilde{\boldsymbol{x}} = (\widetilde{x}_1, \widetilde{x}_2, \cdots, \widetilde{x}_{d_2})$ is a phase vector in $d_2$ dimensions after projection. Define a function $g$ on $d_2$ dimension as

$$\begin{aligned} g(\widetilde{\boldsymbol{x}}) &:= f(\cdots, M^{R(\ell-1,d_1)}\widetilde{x}_{Q(\ell-1,d_1)+1}, \cdots) \\ &= \sum_{\boldsymbol{n}\in\mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \sum_{q=0}^{d_2-1}\left(\sum_{r=0}^{d_1-1} n_{(d_1 q+r+1)} M^r\right)\widetilde{x}_{q+1}}, \end{aligned} \tag{2.26}$$

where $M^{R(\ell-1,d_1)}\widetilde{x}_{Q(\ell-1,d_1)}$ is the $\ell^{\text{th}}$ element of the input of $f$. If we project frequency vectors of $g$ onto the $\widetilde{k}^{\text{th}}$ axis, then the $k^{\text{th}}$ element of a frequency vector $\widetilde{\boldsymbol{n}}$ can be found in the following computation,

$$\begin{aligned} \boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k} &= \Big(g(0\boldsymbol{e}_{\widetilde{k}} + \epsilon\boldsymbol{e}_k), g(\frac{1}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon\boldsymbol{e}_k), \cdots, g(\frac{p-1}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon\boldsymbol{e}_k)\Big) \\ \widetilde{n}_k &= \frac{1}{2\pi\epsilon}\mathrm{Arg}\Big(\frac{\mathcal{F}(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k})[h]}{\mathcal{F}(\boldsymbol{g}_p^{\widetilde{k}})[h]}\Big) \\ c_{\widetilde{\boldsymbol{n}}} &= \frac{1}{p}\mathcal{F}(\boldsymbol{g}_p^{\widetilde{k}})[h], \end{aligned} \tag{2.27}$$

where $\boldsymbol{e}_{\widetilde{k}}$ is the $\widetilde{k}^{\text{th}}$ unit vector with length $d_2$, i.e., all elements are zero except the $\widetilde{k}^{\text{th}}$ one with entry 1. The last two equalities in (2.27) hold as long as $\widetilde{n}_{\widetilde{k}}$ is the only one congruent to $h$ modulo $p$ among all $\widetilde{k}^{\text{th}}$ elements of the frequency vectors, and $\widetilde{\boldsymbol{n}}$ does not collide with any other frequency vector due to the projection onto the $\widetilde{k}^{\text{th}}$ axis. The test for checking whether these conditions are satisfied is

$$\frac{|\mathcal{F}(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k})[h]|}{|\mathcal{F}(\boldsymbol{g}_p^{\widetilde{k}})[h]|} = 1 \tag{2.28}$$

for all $1 \leq k \leq d_2$. The projections onto the $\widetilde{k}^{\text{th}}$ axis, where $\widetilde{k} = 1, \cdots, d_2$, take turns until we recover all frequency vectors and their coefficients. After that we wrap the unwrapped frequency vectors up from $d_2$ to $D$ dimension. Since the unwrapping transformation is one-to-one, this inverse transformation is well-defined.

So far, we assumed that the dimension $D$ can be divided into two integers, $d_1$ and $d_2$. For the case that $D$ is a prime number, or both $d_1$ and $d_2$ are so large that the unwrapped data has a bandwidth such that $\epsilon$ is below the machine precision, a strategy of divide and conquer can be applied. In that case we can think about applying partial unwrapping method in a way that each unwrapped component has a different size of bandwidth. If $D$ is 3, for example, then we can unwrap the first two components of the frequency vector onto one dimension and the last one lies in the same dimension. In that case, the unwrapped data is in two dimensions, and the bandwidth of the first component is bounded by $M^2$ and that of second component is bounded by $M$. In this case we can choose a shift $\epsilon < 1/M^2$ where $M^2$ is the largest bandwidth. We can extend this to the general case, so the partial unwrapping method has a variety of choices balancing the bandwidth and machine precision.

### 2.3.2 Probability of Worst Case Scenario

In this section, we give an upper bound of the probability of the worst case assuming that we randomly choose a partial unwrapping method. As addressed in the Section 2.3.1, there is flexibility in choosing certain partial unwrapping method. Assuming a certain partial unwrapping method and considering a stronger condition to avoid its failure, we can find the upper bound of the probability of the worst case where there is a collision for each parallel projection.

For simple explanation, consider a two dimensional problem. Choosing the first frequency vector $\boldsymbol{n}_1 = (n_{11}, n_{12})$ on a two dimensional plane, if the second frequency vector, $\boldsymbol{n}_2 = (n_{21}, n_{22})$, is not on the vertical line crossing $(n_{11}, 0)$ and the horizontal line crossing $(0, n_{12})$, then the projection method works. Then if the third frequency vector is not on four lines, those two lines mentioned before, the vertical line crossing $(n_{21}, 0)$ and the horizontal line crossing $(0, n_{22})$, then again the projection method works. We keep choosing next frequency vector in this way, excluding the lines containing previous frequencies. Thus, letting such event $A$, the probability that the projection method fails is bounded above by $1-\mathbb{P}(A)$.

Generally, let us assume that we randomly choose a partial unwrapping, without loss of generality, the total dimension is $D = d_1+d_2+\cdots+d_r$ where $r$ is the number of subspaces and $d_1, d_2, \cdots, d_r$ are the dimensions of each subspace. That is, partially unwrapped frequency vectors are in $r < D$ dimensions and each bandwidth is $M^{d_1}, M^{d_2}, \cdots, M^{d_r}$, respectively, which is an integer strictly larger than 1. Then, the failure probability of projection method is bounded above by

$$\begin{aligned}
1-\prod_{j=1}^{s}\mathbb{P}(A_j) &\le 1-\prod_{j=1}^{s}\frac{M^D-(j-1)(M^{d_1}+M^{d_2}+\cdots+M^{d_r})}{M^D} \\
&= 1-\frac{1}{\tau^s}\frac{\tau!}{(\tau-s)!} \qquad \left(\tau := \frac{M^D}{M^{d_1}+M^{d_2}+\cdots+M^{d_r}}\right) \\
&\sim 1-\frac{1}{\tau^s}\sqrt{\frac{\tau}{\tau-s}}\frac{\left(\frac{\tau}{e}\right)^{\tau}}{\left(\frac{\tau-s}{e}\right)^{\tau-s}} \qquad \text{(by Sterling's formula)} \\
&= 1-\frac{1}{e^s}\left(1-\frac{s}{\tau}\right)^{-\frac{\tau}{s}\cdot s}\left(1-\frac{s}{\tau}\right)^{s-\frac{1}{2}} \qquad (2.29)
\end{aligned}$$

where $A_j$ is the event that we choose $j$th frequency not on the lines, crossing formerly chosen frequency vectors and parallel to each coordinate axis. Noting $M^D = M^{d_1}\times M^{d_2}\times\cdots\times M^{d_r}$, sparsity $s$ is relatively small compared to $M^D$, and $\tau$ is large, we can see that the upper bound above gets closer to 0 as $D$ or $M$ grows to infinity.

**Algorithm 3** Multidimensional Sparse Fourier Algorithm Pseudo Code

```
 1: procedure MultiPhaseshift
 2: Input: $f, C, s, M, D, d_1, d_2, \epsilon$
 3: Output: $R$
 4:     $R \leftarrow \emptyset$
 5:     $t \leftarrow 1$
 6:     while $|R| < s$ do
 7:         $s^* \leftarrow s - |R|$
 8:         $p \leftarrow t^{\text{th}}$ prime number $\geq Cs^*$
 9:         $\widetilde{k} \leftarrow (t \bmod d_2) + 1$
10:         $Q(\widetilde{\boldsymbol{x}}) = \sum_{(\widetilde{\boldsymbol{n}}, c_{\widetilde{\boldsymbol{n}}}) \in R} c_{\widetilde{\boldsymbol{n}}} e^{2\pi i \widetilde{\boldsymbol{n}} \cdot \widetilde{\boldsymbol{x}}}$
11:         for $k = 1 \rightarrow d_2$ do
12:             for $\ell = 0 \rightarrow p - 1$ do
13:                 $f_{p,\epsilon}^{\widetilde{k},k}[\ell] = f(\sum_{j=0}^{d_1-1} M^j \frac{\ell}{p} \boldsymbol{e}_{d_1(\widetilde{k}-1)+j} + \epsilon \sum_{j=0}^{d_1-1} M^j \boldsymbol{e}_{d_1(k-1)+j}) - Q(\frac{\ell}{p} \boldsymbol{e}_{\widetilde{k}} + \epsilon \boldsymbol{e}_k)$
14:                 $f_p^{\widetilde{k}}[\ell] = f(\sum_{j=0}^{d_1-1} M^j \frac{\ell}{p} \boldsymbol{e}_{d_1(\widetilde{k}-1)+j}) - Q(\frac{\ell}{p} \boldsymbol{e}_{\widetilde{k}})$
15:             end for
16:             $\mathcal{F}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k}) = \text{FFT}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k})$
17:             $\mathcal{F}(\boldsymbol{f}_p^{\widetilde{k}}) = \text{FFT}(\boldsymbol{f}_p^{\widetilde{k}})$
18:             $\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}}) = \text{SORT}(\mathcal{F}(\boldsymbol{f}_p^{\widetilde{k}}))$
19:         end for
20:         for $h = 0 \rightarrow s^* - 1$ do
21:             $\ell \leftarrow 0$
22:             for $k = 1 \rightarrow d_2$ do
23:                 if $\left| \frac{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}})[h]|}{|\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{p,\epsilon}^{\widetilde{k},k})[h]|} - 1 \right| < \epsilon$ then
24:                     $\ell \leftarrow \ell + 1$
25:                 end if
26:                 $\widetilde{n}_k = \frac{1}{2\pi\epsilon} \text{Arg}\left( \frac{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{\mathrm{p},\epsilon}^{\widetilde{\mathrm{k}},\mathrm{k}})[\mathrm{h}]}{\mathcal{F}^{\text{sort}}(\boldsymbol{f}_{\mathrm{p}}^{\widetilde{\mathrm{k}}})[\mathrm{h}]} \right)$
27:                 $c_{\widetilde{\boldsymbol{n}}} = \frac{1}{p} \mathcal{F}^{\text{sort}}(\boldsymbol{f}_p^{\widetilde{k}})[h]$
28:             end for
29:             if $\ell == d2$ then
30:                 $R \leftarrow R \cup (\widetilde{\boldsymbol{n}}, c_{\widetilde{\boldsymbol{n}}})$
31:             end if
32:         end for
33:         prune small coefficients from $R$
34:         $t \leftarrow t + 1$
35:     end while
36:     inverse-transform each $\widetilde{\boldsymbol{n}}$ in $d_2$-D to $\boldsymbol{n}$ $D$-D and restore it in $R$
37: end procedure
```

## 2.4 Analysis

In this section, we analyze the performance of our algorithms suggested. We will prove that the tilting method works well in two dimensions but explain that it is hard to extend the idea to the general high dimensional setting. However, it was shown in Section 2.3.2 that the probability of the worst-case scenario is extremely small. Furthermore, the average-case runtime and sampling complexity is shown in this section under the assumption that there are no frequency vectors forming a worst-case scenario. The following theorem shows that the tilting method in 2D recovers all frequency pairs even though they form a worst-case scenario.

**Theorem 1.** *Let $\boldsymbol{n} = (n_1, n_2) \in \mathcal{S} \subset \left[-\frac{M}{2}, \frac{M}{2}\right)^2 \cap \mathbb{Z}^2$. If $\tan\theta = \frac{a}{b}$ such that $c > b > a$ are Pythagorean triples where $b > 2M$ and $a$ are relative primes, then all $(cn_1\cos\theta - cn_2\sin\theta, cn_1\sin\theta + cn_2\cos\theta)$ rotated by $\theta$ does not collide with any other pair through the parallel projection. Thus, all rotated pairs can be identified by the parallel projection method.*

*Proof.* Suppose that any two frequency pairs $\boldsymbol{n} = (n_1, n_2)$ and $\boldsymbol{n'} = (n'_1, n'_2)$ cannot be recovered through the tilting method. This implies that the slope of the line crossing $\boldsymbol{n}$ and $\boldsymbol{n'}$ is perpendicular to $\tan\theta$, and thus those two pairs collide at least once if they are projected onto each principal axis rotated by the angle $\theta$. This results in the fact that $\frac{n_2-n'_2}{n_1-n'_1}$ is either $\frac{a}{b}$ or $-\frac{b}{a}$. However, this is a contradiction since $-M < n_1 - n'_1,\ n_2 - n'_2 < M$, and $a$ and $b > 2M$ are relative primes. $\square$

Theorem 1 implies that all frequency pairs can be distinguished by the tilting method with only one proper angle $\theta$. It is natural to think about extending this idea to the high-dimensional setting. However, it is not as easy to find a proper slope as in two dimensions using the Pythagorean triples. Moreover, even though we consider choosing a finite set of random angles guaranteeing that it includes the proper one, each line crossing between two arbitrary vectors in general $D$ dimensions has infinitely many lines perpendicular to it so that such a finite set is difficult to find again. It should be noted that there is a variety of the worst-case scenario where the relatively simple tilting method still works. For example, if the frequency vectors are located on the two dimensional subspace, then the tilting method applied on the subspace would recover all the frequencies. There still exist trickier worst-case scenarios, however, such that the frequencies form a $D$-dimensional cube or more complicated structures. It is still worth exploring further about the tilting method. On the other hand, we also have shown in Section 2.3.2 that the worst-case scenario rarely happens in extremely high dimensions.

In order to obtain the average-case runtime and sampling complexity of the parallel projection method under the assumption that there is no worst-case scenario, we utilize the probability recurrence relation as in [1]. We remind readers that each entry of the frequency vectors should be distinguished modulo $M$ due to parallel projection as well as be distinguished modulo $p$. Since $C = 5$ determining $p$ is shown in [1] to ensure that 90% of frequencies are isolated modulo $p$ on average, at least 90% of frequencies are isolated modulo $M$ on average if $M \geq p$. Taking the union bound of failure probabilities of isolation modulo $p$ and $M$ yields at least 80% of frequencies are isolated both modulo $M$ and $p$ at the same time. Algorithm 3 with $d_1 = 1$ and $d_2 = D$ implements the direct parallel projection method, and it takes $a(s) = \Theta(Ds \log s)$, the runtime on input of size $s$ and $m(s) = s/5$, the average size of the subproblem. Then, the straightforward application of Theorem 2 in [1] gives the following theorem.

**Theorem 2.** *Assume $M \geq 5s$ and there is no worst-case scenario. Let $T(s)$ denote the runtime of Algorithm 3 on a random signal setting with $d_1 = 1$ and $d_2 = D$. Then $\mathbb{E}[T(s)] = \Theta(Ds \log s)$ and*

$$\mathbb{P}[T(s) > \Theta(Ds \log s) + tDs \log s] \leq 5^{-t}.$$

In a similar manner, Algorithm 3 takes $a(s) = \Theta(Ds)$, the number of samples on input of size $s$ and $m(s) = s/5$, the average size of the subproblem. Thus, Theorem 2 in [1] again gives,

**Theorem 3.** *Assume $M \geq 5s$ and there is no worst-case scenario. Let $S(s)$ denote the number of samples used in Algorithm 3 on a random signal setting with $d_1 = 1$ and $d_2 = D$. Then $\mathbb{E}[S(s)] = \Theta(Ds)$ and*

$$\mathbb{P}[S(s) > \Theta(Ds) + tDs] \leq 5^{-t}.$$

## 2.5 Empirical Result

The partial unwrapping method is implemented in the C language. The pseudocode of this algorithm is shown in Algorithm 3. It is explained in detail in Section 2.5.1. In our experiment, dimension $D$ is set to 100 and 1000, $d_1$ is 5 and $d_2$ is 20 and 200, accordingly. Frequency bandwidth $M$ in each dimension is 20 and sparsity $s$ varies as $1, 2, 2^2, \cdots, 2^{10}$. The value of $\epsilon$ for shifting is set to $1/2M^{d_1}$ and the constant number $C$ determining the prime number $p$ is set to 5.

We randomly choose $s$ frequency vectors $\boldsymbol{n} \in [-M/2, M/2)^D \cap \mathbb{Z}^D$ and corresponding coefficients $c_{\boldsymbol{n}} = e^{2\pi i \theta_{\boldsymbol{n}}} \in \mathbb{C}$ from randomly chosen angles $\theta_{\boldsymbol{n}} \in [0, 1)$ so that the magnitude of each $c_{\boldsymbol{n}}$ is 1. For each $D$ and $s$ we have 100 trials. We get the result by averaging $l_2$ errors, the number of samples used and CPU TICKS out of 100 trials.

Since it is difficult to implement high dimensional FFT and there is no practical high dimensional sparse Fourier transform with wide range of $D$ and $s$ at the same time it is hard to compare the result of ours with others, as so far no one else was able to do FFT on this large data set. Thus we cannot help but show ours only. From Figure 2.4 we can see that the average $l_2$ errors are below $2^{-52}$. Those errors are from all differences of frequency vectors and coefficients of the original and recovered values. Since all frequency components are integers and thus the least difference is 1, we can conclude that our algorithm recover the frequency vectors perfectly. Those errors are only from the coefficients. In Figure 2.5 the average sampling complexity is shown. We can see that the logarithm of the number of samples is almost proportional to that of sparsity. Note that the traditional FFT would show the same sampling complexity even though sparsity $s$ varies since it only depends on the bandwidth $M$ and dimension $D$. In Figure 2.6 the average CPU TICKS are shown. We can see that the logarithm of CPU TICKS is also almost proportional to that of sparsity. Note that the traditional FFT might show the same CPU TICKS even though sparsity $s$ varies since it also depends on the bandwidth $M$ and dimension $D$ only.

### 2.5.1 Algorithm

In this section, the explanation of Algorithm 3 is given. In [1] several versions of 1D algorithms are shown. Among them, non-adaptive and adaptive algorithms are introduced where the input function $f$ is not modified throughout the whole iteration, and is modified by subtracting the function constructed from the data in registry $R$, respectively. In our multidimensional algorithm, however, the adaptive version is mandatory since excluding the contribution of the currently recovered data is the key of our algorithm to avoid the collision of frequencies through projections, whose simple pictorial description is given in Figure 2.2. In Algorithm 3, the function $Q$ is the one constructed from the data in the registry $R$.

Our algorithm begins with entering inputs, a function $f$, a constant number $C$ determining $p$, a sparsity $s$, a bandwidth $M$ of each dimension, a dimension $D$, factors $d_1$ and

$d_2$ of $D$ and a shifting number $\epsilon \leq 1/M^{d_1}$. For each iteration of the algorithm, the number of frequencies to find is updated as $s^* = s - |R|$. It stops when $|R|$ becomes equal to the sparsity $s$. The prime number $p$ is determined depending on this new $s^*$ as $p \geq Cs^*$ and is chosen as the next larger prime number. The lines 13 and 14 of Algorithm 3 represent the partial unwrapping, and sampling with and without shifting from the function where the contribution of former data is excluded. After applying the FFT on each sequence, sorting them according to the magnitude of $\mathcal{F}(\widetilde{\boldsymbol{f}_p^k})$, we check the ratio between the FFT's of the unshifted and shifted sequences to determined whether there is a collision, either from modulo $p$ or a parallel projection. If all tests are passed, then we find each frequency component and corresponding coefficient for the data that passed and store them in $R$. After several iterations, we find all the data and the final wrapping process gives the original frequency vectors in $D$ dimensions.

### 2.5.2 Accuracy

We assume that there is no noise on the data that we want to recover. Figure 2.4 shows that we can find frequencies perfectly and the $l_2$ error from coefficients are significantly small. This error is what we average out over 100 trials for each $D = 100, 1000$ and $s = 1, 2^1, 2^2, \cdots, 2^{10}$ when $M$ is fixed to 20. The horizontal axis represents the logarithm with base 2 of $s$ and the vertical axis represents the logarithm with base 2 of the $l_2$ error. It is increasing as the sparsity $s$ is increasing since the number of nonzero coefficients increases. The red graph in the Figure 2.4 shows the error when the number of dimensions is 100 and the blue one shows the error when the number of dimensions is 1000. Thus, we see that the errors are not substantially impacted by the dimensions.

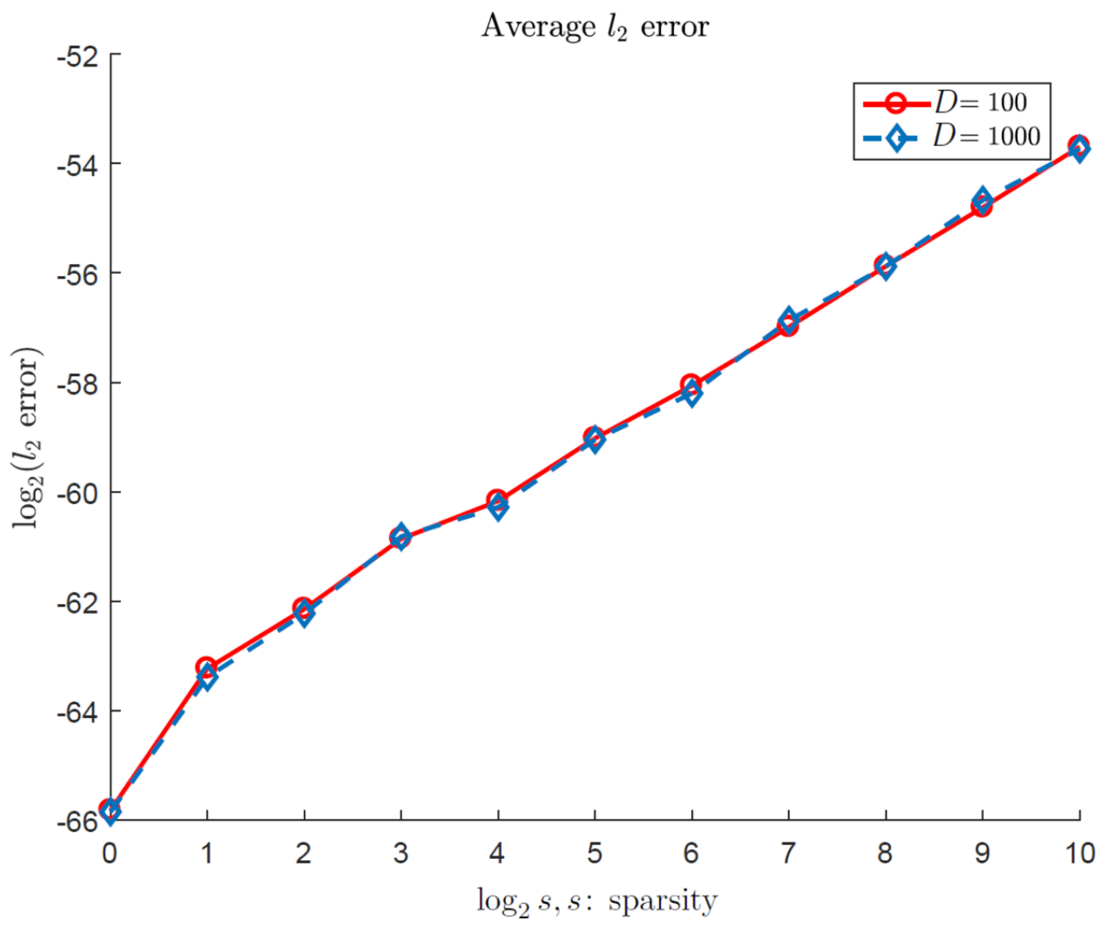


Figure 2.4: Average $l_2$ error

### 2.5.3 Sampling Complexity

Figure 2.5 shows the sampling complexity of our algorithm averaged out from 100 tests for each dimension and sparsity. The horizontal axis means the logarithm with base 2 of $s$ and the vertical axis represents the logarithm with base 2 of the total number of samples from the randomly constructed function which are used to find all frequencies and coefficients. The red graph in the Figure 2.5 shows the sampling complexity when the number of dimensions is 100 and the blue one shows the one when the number of dimensions is 1000. Both graphs increase as $s$ increases. When $D$ is large, we see that it requires more samples since there are more frequency components to find. From the graphs, we see that the scaling seems to

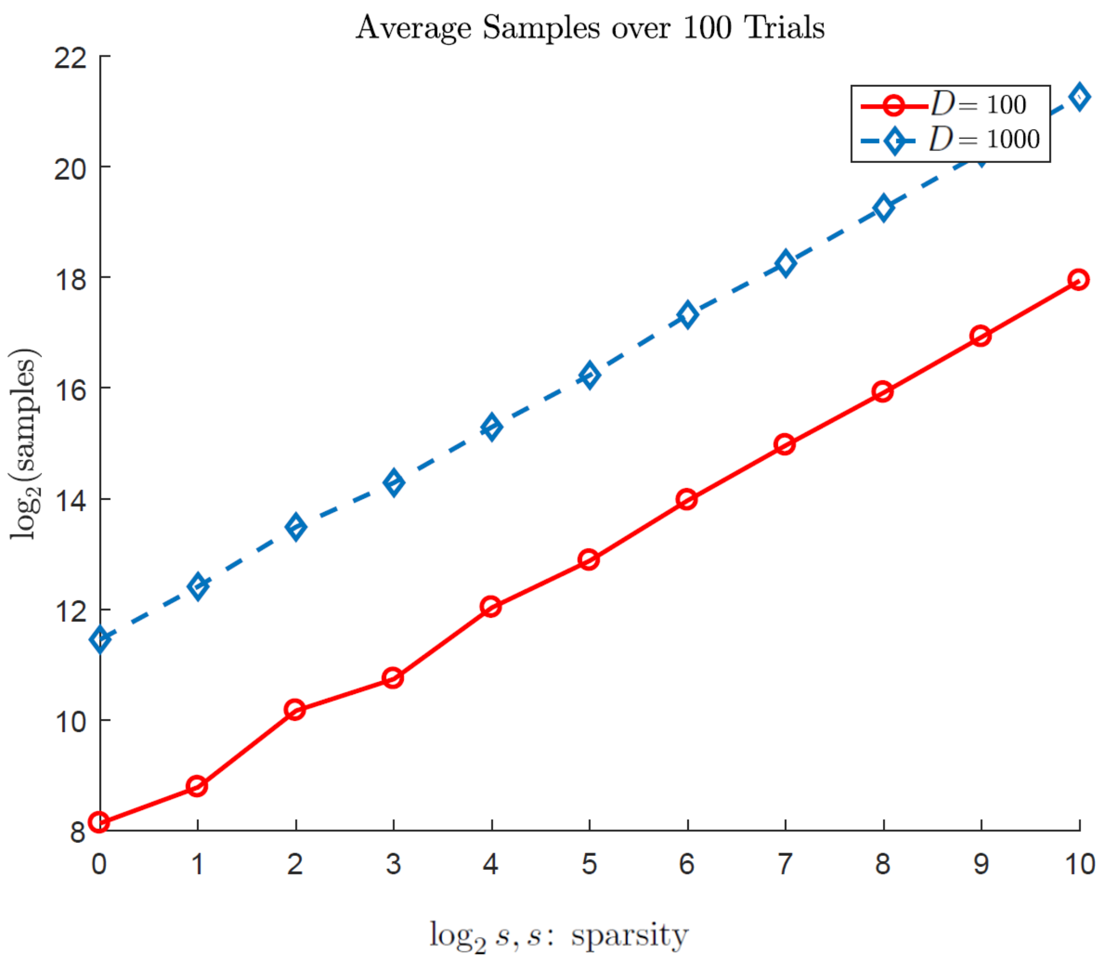


Figure 2.5: Average sampling complexity

be proportional to $D$.

### 2.5.4 Runtime Complexity

In Figure 2.6, we plot the runtime complexity of the main part of our algorithm averaged over 100 tests for each dimension and sparsity. "Main part" means that we have excluded the time for constructing a function consisting of frequencies and coefficients and the time associated with getting samples from it. The horizontal axis is the logarithm, base 2, of $s$ and the vertical axis is the logarithm, base 2, of CPU TICKS. The red curve shows the runtime when we set the number of dimensions to 100 and the blue one shows the same

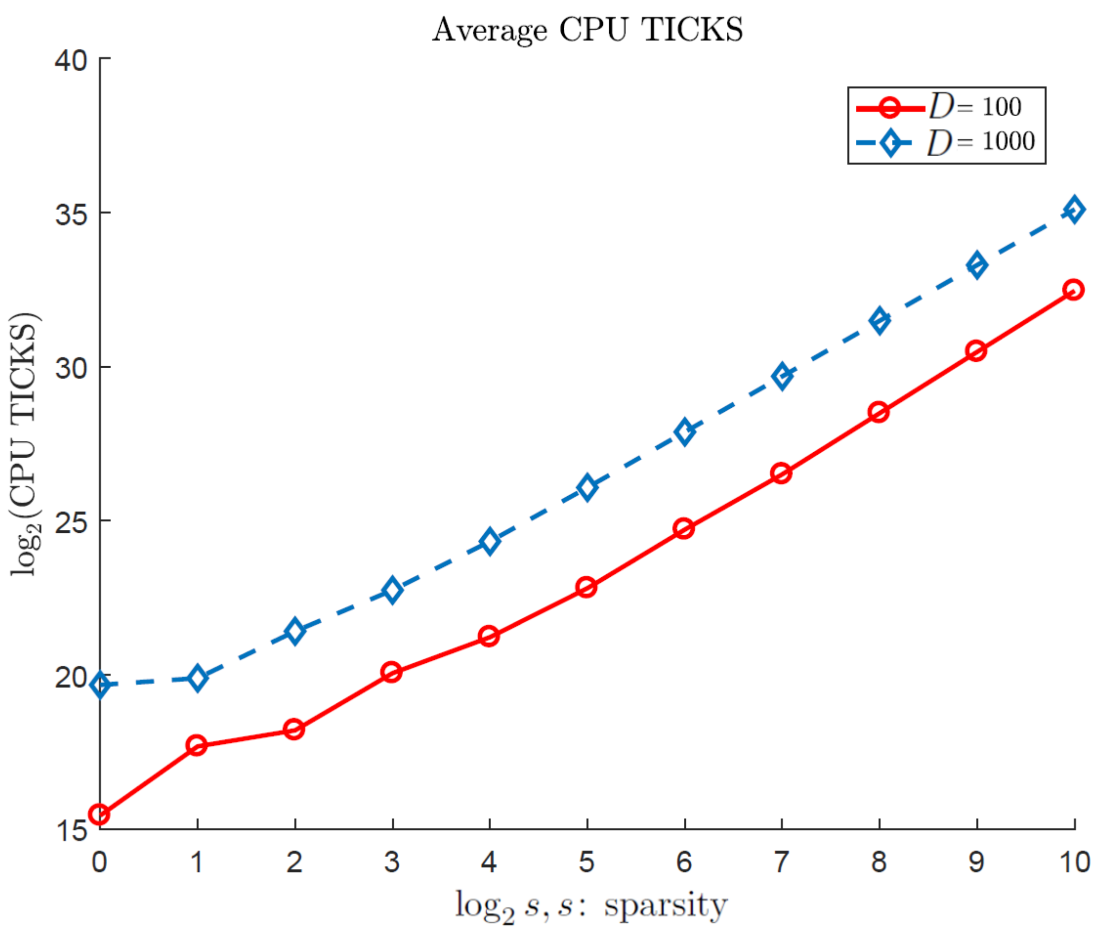


Figure 2.6: Average CPU TICKS

thing when the number of dimensions to 1000. Both plots increase as $s$ increases. When $D$ is larger, the plots show that it takes more time to run the algorithm. From the graphs we see that the runtime looks proportional to $D$.

Unfortunately, the sampling process of getting the samples from continuous functions dominates the runtime of the whole algorithm instead of the main algorithm. To show the runtime of our main algorithm, however, we showed CPU TICKS without sampling process. Reducing the time for sampling is still a problem. In [10] a one-dimensional fully discrete Fourier transform is introduced that we expect to use to reduce it. Exploring how to use this will be one part of our future work.

# Chapter 3

# Multiscale Sparse Fourier Transforms

## 3.1 Preliminaries

### 3.1.1 Notation and Review

In this section, we introduce the notation used throughout this chapter. Let $s$, $D$ and $M$ be natural numbers, $s \ll M^D$, and $\mathcal{D} := [0,1)^D$. We consider a function $f : \mathcal{D} \to \mathbb{C}$ which is $s$-sparse in the $D$-dimensional Fourier domain as follows

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} e^{2\pi i \boldsymbol{n} \cdot \boldsymbol{x}}$$

where each $\boldsymbol{n} \in [-M/2, M/2)^D \cap \mathbb{Z}^D$ and $c_{\boldsymbol{n}} \in \mathbb{C}$. We note that $f$ can be regarded as a periodic function defined on $\mathbb{R}^D$. The aim of sparse Fourier algorithms is to rapidly reconstruct a function $f$ using a small number of its samples. In Chapter 2, the methods were introduced using several different transformations and parallel projections along coordinate axes of frequencies in order to exploit the one-dimensional sparse Fourier algorithm from [1]. These transformations such as partial unwrapping and tilting methods are introduced in order to change the locations of energetic frequencies when the current energetic frequencies are hard or impossible to find through the parallel projections directly. Through those manipulations, each frequency vector $\boldsymbol{n}$ is recovered in an entry-wise fashion. In this way, the linear dependence of runtime and sampling complexities on the dimension $D$ could be shown empirically, which is a great improvement when compared to the $D$-dimensional FFT with the exponential dependence on $D$. The transformations and projections occur in the physical domain, which provides the separation of the frequency vectors in the Fourier domain. That is, each $\boldsymbol{n}$ is transformed to $\boldsymbol{n}'$ and then projected onto several lines, and these can be done by manipulating the sampling points in the physical domain $\mathcal{D}$. Let $u : \mathcal{D}' \to \mathcal{D}$ represent those transformations where $\mathcal{D}'$ is $D$ or less dimensional space and is determined by each transformation. We assume that $\mathcal{D}'$ has $D'$ dimensions. A new function $g : \mathcal{D}' \to \mathbb{C}$ is defined as a composition of $f$ and $u$, i.e.,

$$g(\boldsymbol{x}) := f(u(\boldsymbol{x})).$$

We note that $g$ is still $s$-sparse in the $D'$-dimensional Fourier domain, $[-M'/2, M'/2)^{D'} \cap \mathbb{Z}^{D'}$, whose bandwidth $M'$ depends on each transformation. For example, consider a 4-dimensional function $f$ with the Fourier domain, $[-M/2, M/2)^4 \cap \mathbb{Z}^4$. If a partial unwrapping is applied to $f$ which unwraps each $\boldsymbol{n} = (n_1, n_2, n_3, n_4)$ to $\boldsymbol{n}' = (n_1' + Mn_2', n_3' + Mn_4')$ then it implies that $u : (x_1, x_2) \to (x_1, Mx_1, x_2, Mx_2)$ and $g$ has the Fourier domain, $[-M^2/2, M^2/2)^2 \cap \mathbb{Z}^2$ where accordingly $M' = M^2$ and $D' = 2$. This is one example and there are variations of partial unwrapping methods and tilting methods which can be found in Chapter 2. Using samples of $g$(or $f$), the transformed frequency vectors $\boldsymbol{n}'$ and corresponding Fourier coefficients $c_{\boldsymbol{n}'}(=c_{\boldsymbol{n}})$ are found through the parallel projection method, and $\boldsymbol{n}'$ are transformed back to $\boldsymbol{n}$. Now, we remind the readers using comprehensive notations how $\boldsymbol{n}' = (n_1', n_2', \cdots, n_{D'}')$ can be recovered element-wise using the parallel projection method and the ideas from the one-dimensional sparse Fourier algorithm. Let $p$ be a prime number greater than a constant multiple of $s$, i.e., $p > Cs$ for some constant $C$, $\boldsymbol{n}$ be a fixed frequency vector in $\boldsymbol{\mathcal{S}}$, $k$ be fixed among the set $\{1, 2, \cdots, D'\}$ and $\boldsymbol{e}_k$ be a vector with all zero entries but 1 at the index $k$. Furthermore, $\epsilon$ is defined as a positive number $\leq 1/M'$. To recover the $k^{\text{th}}$ element of each $\boldsymbol{n'}$, we use two sets of $p$-length equispaced samples $\boldsymbol{g}_p^{\widetilde{k}}$ and $\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}$ as follows,

$$\boldsymbol{g}_p^{\widetilde{k}}[\ell] := g\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}}\right) \quad \text{and} \quad \boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}[\ell] := g\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon\boldsymbol{e}_k\right)$$

where $\ell = 0, 1, \cdots, p-1$ and $\widetilde{k} \in \{1, 2, \cdots, D'\}$ is the index of coordinate axis where a particular $\widetilde{\boldsymbol{n}}$ has $\widetilde{k}^{\text{th}}$ element, $\widetilde{n}_{\widetilde{k}}$, different from the $\widetilde{k}^{\text{th}}$ elements of any other energetic frequency vectors. In this case, we refer to the above phenomenon as "no collision from projection". At the same time, if there is no *collision modulo* $p$, i.e., $\widetilde{n}_{\widetilde{k}}$ has the unique remainder modulo $p$ from others then the discrete Fourier transform of each sample set is

$$\begin{aligned}
\mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h] &= p \sum_{n'_{\widetilde{k}} \equiv h \bmod p} c_{\boldsymbol{n'}} = p c_{\widetilde{\boldsymbol{n}}}, \qquad \text{and} \\
\mathcal{F}\left(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}\right)[h] &= p \sum_{n'_{\widetilde{k}} \equiv h \bmod p} c_{\boldsymbol{n'}} e^{2\pi i n'_k \epsilon} = p c_{\widetilde{\boldsymbol{n}}} e^{2\pi i \widetilde{n}_k \epsilon},
\end{aligned} \tag{3.1}$$

respectively. The equations above in (3.1) give a unique entry for the $k^{\text{th}}$ element assuming there does not exist a collision modulo $p$ of the vetor projected onto the $k^{\text{th}}$ axis. Hence, in the above equations, when the second equalities hold, we can recover $c_{\widetilde{\boldsymbol{n}}}$ and $\widetilde{n}_k$ as follows,

$$\widetilde{n}_k = \frac{1}{2\pi\epsilon}\text{Arg}\left(\frac{\mathcal{F}\left(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h]}\right) \qquad \text{and} \qquad c_{\widetilde{\boldsymbol{n}}} = \frac{\mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h]}{p}. \tag{3.2}$$

The right choice of the branch and the shift size $\epsilon \leq \frac{1}{M'}$ make it possible to find the correct $\widetilde{n}_k$. Algorithmically, the two kinds of collisions are guaranteed not to happen using the following test:

$$\left|\frac{\mathcal{F}\left(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h]}\right| = 1, \tag{3.3}$$

which is inspired by the test used in [1]. Practically, we put some threshold $\tau > 0$ so that if the difference between the left and right-hand sides in (3.3) is less than $\tau$, then we conclude that there are no collisions of both kinds. In this case, each $\widetilde{n}_k$ for $k = 1, 2, \cdots, D'$ can be recovered using a pair of sets $\boldsymbol{g}_p^{\widetilde{k}}$ and $\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}$, respectively. Otherwise, we take another prime number for sample length, switch the index of coordinate axis for the projection, update the samples by eliminating the influence from previously found Fourier modes and repeat our procedure as before. Switching the coordinate axis and updating samples reduce the occasions of *collisions from projection*. In Chapter 2, moreover, it is proved that the probability is very low when $D'$ is large that all remaining frequency vectors have *collisions from projection* onto all coordinate axes, which we call the "*worst case scenario*". In the "*worst case scenario*", the parallel projections we just used do not work and thus, we need a rotation mapping, $u'$, defining another function $g' = f \circ u'$.

### 3.1.2 Noise Model

In practice, the samples from $f$(or $g$) are often corrupted by noise. The high-dimenisonal sparse Fourier algorithm introduced in Section 3.1.1 works well when $f$ is exactly $s$-sparse and the samples from $f$ are not noisy. It is not robust to noise since in order to find entries of energetic frequency vectors we compute the fraction $\mathcal{F}\left(\boldsymbol{g}_{p;\epsilon}^{\widetilde{k},k}\right)[h]/\mathcal{F}\left(\boldsymbol{g}_{p}^{\widetilde{k}}\right)[h]$ which is sensitive to noise. In the remaining sections of Chapter 3, we consider the case of noisy measurements of $g$ as follows

$$\boldsymbol{r}_p^{\widetilde{k}}[\ell] := \boldsymbol{g}_p^{\widetilde{k}}[\ell] + z_\ell = g\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}}\right) + z_\ell$$

for $\ell = 0, 1, \cdots, p-1$ where $\boldsymbol{z} := (z_0, z_1, \cdots, z_{p-1})$ is a complex Gaussian random variable with mean $\mathbf{0}$ and variance $\sigma^2 I$. If we apply DFT to this sample set, we get

$$\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h] = \mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h] + \sum_{\ell=0}^{p-1} z_\ell e^{-2\pi i h \frac{\ell}{p}} \tag{3.4}$$

Since $z_\ell$ are i.i.d Gaussian variables, the expectation and variance of the second term in (3.4) are

$$\mathbb{E}\left[\sum_{\ell=0}^{p-1} z_\ell e^{-2\pi i h \frac{\ell}{p}}\right] = 0$$

and

$$\operatorname{Var}\left[\sum_{\ell=0}^{p-1} z_\ell e^{-2\pi i h \frac{\ell}{p}}\right] = p\sigma^2,$$

respectively. Accordingly,

$$\mathbb{E}\left[\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]\right] = \mathcal{F}\left(\boldsymbol{g}_p^{\widetilde{k}}\right)[h]$$

and

$$\mathrm{Var}\left[\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]\right] = p\sigma^2.$$

For noisy shifted samples set $\boldsymbol{r}_{p;\epsilon}^{\widetilde{k},k} := \boldsymbol{g}_{p;\epsilon}^{\widetilde{k},k} + \widetilde{\boldsymbol{z}}$ with an i.i.d Gaussian random vector $\widetilde{\boldsymbol{z}}$, we have likewise

$$\mathbb{E}\left[\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h]\right] = \mathcal{F}\left(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}\right)[h]$$

and

$$\mathrm{Var}\left[\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h]\right] = p\sigma^2.$$

In the case of $n'_{\widetilde{k}}$ not having collisions both from projection and modulo $p$, and $n'_{\widetilde{k}} \equiv h \pmod{p}$,

$$\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h] = pc_{\boldsymbol{n}'} + \mathcal{O}(\sigma\sqrt{p}) \tag{3.5}$$

and

$$\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h] = pc_{\boldsymbol{n}'}e^{2\pi i n'_k \epsilon} + \mathcal{O}(\sigma\sqrt{p})$$

for each $k = 1, 2, \cdots, D'$. As a result, we get

$$\frac{\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]} = e^{2\pi i n'_k \epsilon} + \mathcal{O}(\frac{\sigma}{c'_{\boldsymbol{n}}\sqrt{p}}) \tag{3.6}$$

and note that if there were no noise in samples, we only have the first term on the right side of (3.6) which makes it possible to recover $n'_k$ by taking its argument and dividing it by $2\pi\epsilon$ as (3.2). With noisy samples, however, it is corrupted with noise which is a multiple of

$\frac{\sigma}{c'_{\boldsymbol{n}}\sqrt{p}}$. Defining

$$\hat{n}_k := \frac{1}{2\pi\epsilon}\mathrm{Arg}\left(\frac{\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{r}_{p}^{\widetilde{k}}\right)[h]}\right), \tag{3.7}$$

we want to see how far $\hat{n}_k$ is from $n'_k$. For this purpose, we introduce the *Lee norm* associated with a lattice $\mathcal{L}$ in $\mathbb{R}$ as $\|z\|_{\mathcal{L}} := \min_{y\in\mathcal{L}}|z-y|$ for $z \in \mathbb{R}$ and the related property that under the Lee norm associated with the lattice $2\pi\mathbb{Z}$,

$$\|\mathrm{Arg}(\gamma+\nu) - \mathrm{Arg}(\gamma)\|_{2\pi\mathbb{Z}} = \left\|\mathrm{Arg}\left(1+\frac{\nu}{\gamma}\right)\right\|_{2\pi\mathbb{Z}} \leq \frac{\pi}{2}\left|\frac{\nu}{\gamma}\right|, \tag{3.8}$$

where $|\gamma| \geq |\nu|$ with $\gamma, \nu \in \mathbb{C}$. By choosing the sample length $p$ large enough depending on the least magnitude nonzero $c_{\min}$ and the noise level $\sigma$, (3.8) can be applied to (3.6) as follows,

$$\left\|\mathrm{Arg}\left(\frac{\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{r}_{p}^{\widetilde{k}}\right)[h]}\right) - 2\pi n'_k\epsilon\right\|_{2\pi\mathbb{Z}} \leq \mathcal{O}\left(\frac{\sigma}{|c_{\min}|\sqrt{p}}\right).$$

Consequently,

$$\|\hat{n}_k - n'_k\|_{\mathbb{Z}} \leq \mathcal{O}\left(\frac{\sigma}{2\pi\epsilon\ |c_{\min}|\sqrt{p}}\right), \tag{3.9}$$

which implies that the error of our estimate $\hat{n}_k$ to $n'_k$ is controlled by the size of $\frac{\sigma}{\epsilon\ |c_{\min}|\sqrt{p}}$. Thus, $p$ needs to be chosen carefully depending on $\frac{\sigma}{\epsilon\ |c_{\min}|}$. On the other hand, from (3.5), we can approximate the corresponding coefficient $c_{\boldsymbol{n}'}$ with the error of size $\mathcal{O}(\sigma/\sqrt{p})$ as follows

$$c_{\boldsymbol{n}'} = \frac{1}{p}\mathcal{F}\left(\boldsymbol{r}_{p}^{\widetilde{k}}\right)[h] + \mathcal{O}\left(\frac{\sigma}{\sqrt{p}}\right). \tag{3.10}$$

## 3.2 Multiscale Method

In [2], *rounding* and *multiscale* methods for the one-dimensional sparse Fourier algorithm for noisy data were introduced. Both methods use the fact that the peaks of the DFT are robust to relatively high noise, i.e., $h$ can be correctly found such that $n \equiv h(\text{mod } p)$ for an energetic frequency $n$. The *Rounding* method is efficient when $\sigma$ is relatively small, which approximates such $n := bp + h$ for some integer $b$ up to $p/2$ error and rounds a multiple of $p$ in order to get the correct $b$. It was shown that $p \geq \max\{C_1 s, C_2(\frac{\sigma}{\epsilon|c_{\min}|})^{2/3}\}$ for some constants $C_1$ and $C_2$ makes it possible to correctly find $n$ through the *rounding* method. On the other hand, the *multiscale* method was introduced for relatively large $\sigma$. It prevents $p$ from becoming too large, which happens for large $\sigma$ in the rounding method. In this section, we focus on extending the *multiscale* method to recover high dimensional frequencies by gradually fixing each entry estimation with several shifts $\epsilon_q$.

### 3.2.1 Description of Frequency Entry Estimation

Let $\boldsymbol{n}' \in \mathcal{S}'$ and $k \in \{1, 2, \cdots, D'\}$ be fixed. The set $\mathcal{S}'$ is defined to be the set containing all $\boldsymbol{n}'$ transformed from $\boldsymbol{n} \in \mathcal{S}$. The target frequency entry $n'_k$ is assumed not to have collisions both from projection and modulo $p$. We start with a coarse estimation $n'_{k,0}$ of $n'_k$ defined by

$$n'_{k,0} := \frac{1}{2\pi\epsilon_0}\text{Arg}\left(\frac{\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon_0}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]}\right),$$

where $\epsilon_0 \leq 1/M'$. Then $n'_{k,0} \equiv n'_k(\text{mod } p)$ even though it is not guaranteed that $n'_{k,0} = n'_k$. Thus, we need to improve the approximation. With each correction, the solution is improved by $l$ digits where $l$ depends on the parameters that are chosen in the method as well as noise.

Each correction term is calculated with a choice of growing $\epsilon_q > 1/M'$, i.e., $\epsilon_{q-1} < \epsilon_q$ for all $q \geq 1$. With the initialization $b_{k,0} := \epsilon_0 n'_{k,0}$, the correction terms are calculated in the following way,

$$b_{k,q} := \frac{1}{2\pi}\mathrm{Arg}\left(\frac{\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{r}_{p}^{\widetilde{k}}\right)[h]}\right)$$

and

$$n'_{k,q} := n'_{k,q-1} + \frac{\left(b_{k,q} - \epsilon_q n'_{k,q-1}\right)\left(\mathrm{mod}\ \left[-\frac{1}{2},\frac{1}{2}\right)\right)}{\epsilon_q}$$

for $q \geq 1$, where $x \pmod{[-1/2, 1/2)}$ is defined to be the value $a \in [-1/2, 1/2)$ such that $x \equiv a \pmod 1$. Using this fact,

$$b_{k,q} \approx \epsilon_q n'_k \left(\mathrm{mod}\ \left[-\frac{1}{2},\frac{1}{2}\right)\right),$$

the error $n'_k - n'_{k,q-1}$ can be estimated as

$$\begin{aligned}\epsilon_q(n'_k - n'_{k,q-1}) &= \epsilon_q n'_k - \epsilon_q n'_{k,q-1} \\ &\approx (b_{k,q} - \epsilon_q n'_{k,q-1})\ \left(\mathrm{mod}\ \left[-\frac{1}{2},\frac{1}{2}\right)\right)\end{aligned}$$

and thus, in a similar manner to (3.9),

$$\mathcal{O}\left(\frac{\sigma}{\epsilon_q c_{\min}\sqrt{p}}\right) = (n'_k - n'_{k,q-1}) - \frac{\left(b_{k,q} - \epsilon_q n'_{k,q-1}\right)\left(\mathrm{mod}\ \left[-\frac{1}{2},\frac{1}{2}\right)\right)}{\epsilon_q} \tag{3.11}$$

$$= n'_k - \left(n'_{k,q-1} + \frac{\left(b_{k,q} - \epsilon_q n'_{k,q-1}\right)\left(\mathrm{mod}\ \left[-\frac{1}{2},\frac{1}{2}\right)\right)}{\epsilon_q}\right) \tag{3.12}$$

$$= n'_k - n'_{k,q}. \tag{3.13}$$

As the correction is repeated with larger $\epsilon_q$, the error of the estimate decreases. In other words, we approximate $n'_k$ by its most significant bits and the next significant bits repeatedly. The performance of this multiscale method is shown in detail in the next section.

### 3.2.2 Analysis of Multiscale Method

The following theorem shows how the correction term $d_{k,q}/\epsilon_q$ is constructed in each iteration and how large the error of estimate $n'_k$ after $L$ iterations is in the multiscale frequency entry estimation procedure.

**Theorem 4.** *Let $\boldsymbol{n}' \in \mathcal{S}', k \in \{1, 2, \cdots, D'\}$ be fixed and $n'_k \in \left[-\frac{M'}{2}, \frac{M'}{2}\right)$. Let $0 < \epsilon_0 < \epsilon_1 < \cdots < \epsilon_L$ and $b_{k,0}, b_{k,1}, \cdots, b_{k,L} \in \mathbb{R}$ such that*

$$\|\epsilon_q n'_k - b_{k,q}\|_{\mathbb{Z}} < \delta, \qquad 0 \le q \le L$$

*where $0 < \delta < \frac{1}{4}$. Assume that $\epsilon_0 \le \frac{1-2\delta}{M'}$ and $\beta_q := \frac{\epsilon_q}{\epsilon_{q-1}} \le \frac{1-2\delta}{2\delta}$. Then there exist $d_{k,0}, d_{k,1}, \cdots, d_{k,L} \in \mathbb{R}$, each computable from $\{\epsilon_q\}$ and $\{b_{k,q}\}$ such that*

$$|\widetilde{n}_k - n'_k| \le \frac{\delta}{\epsilon_0} \prod_{q=1}^{L} \beta_q^{-1}, \qquad \textit{where} \qquad \widetilde{n}_k := \sum_{q=0}^{L} \frac{d_{k,q}}{\epsilon_q}.$$

*Proof.* The proof is the same as the proof of Theorem 4.2 in [2] since each entry of frequency vectors is corrected in the same way as each one-dimensional frequency $w$ in [2] is. Each $d_q$ is defined as

$$d_0 := b_0 \qquad \text{and}$$

$$d_q := b_q - \epsilon_q \lambda_{q-1} \left( \bmod \left[ -\frac{1}{2}, \frac{1}{2} \right) \right) \qquad \text{for } q \geq 1,$$

where $\lambda_q = d_q / \epsilon_q$ for $q \geq 0$. □

**Corollary 1.** *Assume that we let $\beta_q = \beta$ in Theorem 4 where $\beta \leq \frac{1-2\delta}{2\delta}$, i.e., $\epsilon_q = \beta^q \epsilon_0$ for all $q \geq 1$. Let $p > 0$ and $L \geq \left\lfloor \log_\beta \frac{2\delta}{\epsilon_0} \right\rfloor + 1$. Then,*

$$\left| \widetilde{n}_k - n'_k \right| \leq \frac{\delta}{\epsilon_0} \beta^{-L} < \frac{1}{2}.$$

*Proof.* This is straightforward corollary of Theorem 4. □

Corollary 1 looks very similar to Corollary 4.3 in [2]. However, it is different in that the iteration number is increased in order to make the error between $\widetilde{n}_k$ and $n'_k$ less than 1/2 instead of $p/2$. In [2], the remainder $h$ modulo $p$ of each energetic $n$ is known by sorting out the $s$ largest DFT components of $p$-length unshifted sample vector so that $p/2$ error bound is enough to guarantee the exact recovery of $n$. On the other hand, in the high-dimensional setting of this section, the remainders of all entries of $n'_k$ are not known. Instead, the remainder of $n'_{\widetilde{k}}$ is only known where $\widetilde{k}$ is the index of coordinate axis where the frequencies are projected. Thus, we decrease the error further by enlarging the iteration number $L$ and are able to recover the exact $n'_k$ by rounding when the error is less than 1/2.

**Remark 1.** *Similar to Theorem 4.4 in [2], the admissible size of $\delta$ can be estimated as follows,*

$$\delta = \min \left( \frac{1 - \epsilon_0 M'}{2}, \frac{1}{2\beta + 2} \right) \tag{3.14}$$

*under the assumption $\epsilon_0 \leq \frac{1-2\delta}{M'}$ of Corollary 1 together with the assumption $\beta \leq \frac{1-2\delta}{2\delta}$ of*

*Theorem 4.*

## 3.3 Algorithm

In this section, the parameters that determine the performance of our algorithm are introduced and the way that they are chosen is shown. Those are affected by the noise level, $\sigma$. Furthermore, the pseudocode of the multiscale algorithm and the related explanation are provided. It is important to know how the frequency entry estimation works in the algorithm and how the collision detection tests are modified from the tests in Chapter 2 in order to make them tolerant of noise.

### 3.3.1 Choice of $p$

The sample length $p$ affects the total runtime complexity since the discrete Fourier transform is applied to all sample sets taken to recover the frequency entries. Due to this, we want to make it as small as possible. At the same time, however, we can see from (3.9) that the error between the target entry and its approximation becomes smaller if a larger $p$ is taken. Thus, as discussed in Section 3.2, if $p$ is large enough, then the *rounding method* instead of *multiscale method* can recover the exact frequency by rounding (3.7) to the nearest integer of the form $pv + h$ with an integer $v$. In this case, $p$ is large enough to diminish the influence of $\sigma$, and therefore if $\sigma$ is large, so is $p$. Instead, the *multiscale method* makes it possible to enlarge $p$ moderately. From Theorem 4, we get

$$|n'_{k,q+1} - n'_k| < \frac{\delta}{\epsilon_{q+1}} \tag{3.15}$$

and (3.13) implies

$$|n'_{k,q+1} - n'_k| \leq \mathcal{O}\left(\frac{\sigma}{2\pi\epsilon_q\ |c_{\min}|\sqrt{p}}\right). \tag{3.16}$$

By putting the right side of (3.16) as $C_\sigma \frac{\sigma}{\epsilon_q\ |c_{\min}|\sqrt{p}}$ with some constant $C_\sigma$ and equating both right sides of (3.15) and (3.16), $\beta := \epsilon_{q+1}/\epsilon_q$ can be estimated as

$$\beta = \frac{2\pi\delta\sqrt{p}}{c_{\min}C_\sigma\sigma}. \tag{3.17}$$

$\beta$ determines the choice of $\epsilon_q$ for each iteration, i.e., $\epsilon_q = \beta^q\epsilon_0$ where $\epsilon_0$ is chosen to be less than $1/M'$. Combining (3.14) and (3.17), the sample length $p$ can be calculated as

$$p = \left(\frac{\beta(\beta+1)c_{\min}C_\sigma\sigma}{\pi}\right)^2$$

when $\epsilon_0 = \frac{1}{2M'}$ and $\beta > 1$, which implies $\delta = \frac{1}{2\beta+2}$. Eventually, the sample length $p$ for the multiscale method needs to satisfy

$$p > \max\left\{C_1 s, \left(\frac{\beta(\beta+1)c_{\min}C_\sigma\sigma}{\pi}\right)^2\right\}, \tag{3.18}$$

where $p > C_1 s$ ensures the sample is long enough so that the 90% of all energetic frequencies are not collided modulo $p$ on average which comes from the pigeonhole argument in [1].

### 3.3.2 Collision Detection Tests

As mentioned in Section 3.1.1, frequencies are recovered only when there are no collisions from the projection and the modulo $p$ division. These conditions are satisfied if

$$\left| \left| \frac{\mathcal{F}\left(\boldsymbol{g}_{p,\epsilon}^{\widetilde{k},k}\right)[h]}{\mathcal{F}\left(\boldsymbol{g}_{p}^{\widetilde{k}}\right)[h]} \right| - 1 \right| < \tau, \tag{3.19}$$

for $k = 1, \cdots, D'$ and some small $\tau > 0$ which are the practical tests of (3.3). In our noisy setting, Equation (3.6) implies that the left hand side of (3.19) is bounded above by $\mathcal{O}\left(\frac{\sigma}{c_{\min}\sqrt{p}}\right)$. Thus, we set our threshold $\tau$ as a constant multiple of $\frac{\sigma}{c_{\min}\sqrt{p}}$. Moreover, since we iteratively update the estimates $n'_{k,q}$ for $q = 0, 1, \cdots, L$, we reject the estimate after $L$ iterations if the tests fail for more than $\eta(L+1)$ times for each $k^{\text{th}}$ entry with $k = 1, 2, \cdots, D'$ where $\eta$ is a fraction$< 1$. Numerical experiments indicate $\eta = \frac{1}{4}$ is a good number.

### 3.3.3 Number of Iterations

From Corollary 1, $L \geq \left\lfloor \log_\beta \frac{2\delta}{\epsilon_0} \right\rfloor + 1$ guarantees we get the approximation error, $\left|\widetilde{n}_k - n'_k\right| < \frac{1}{2}$ which is required to recover the exact $n'_k$ by rounding $\widetilde{n}_k$ to the nearest integer. With our choice of $\epsilon_0 = \frac{1}{2M'}$ and the fact that $\delta < 1$, $L = \left\lfloor \log_\beta M' \right\rfloor + 1$ suffices to satisfy the 1/2 error bound. For example, if each $\boldsymbol{n} \in \left[-\frac{M}{2}, \frac{M}{2}\right)^D \cap \mathbb{Z}^D$ is partially unwrapped to some value in $\left[-\frac{M^{d_1}}{2}, \frac{M^{d_1}}{2}\right)^{d_2} \cap \mathbb{Z}^{d_2}$ where $d_1$ and $d_2$ are positive integers satisfying $D = d_1 d_2$, then $d_2$ entries of each energetic frequency are recovered element-wisely after $L = \mathcal{O}(d_1 \log M)$ iterations.

### 3.3.4 Description of Our Pseudocode

In this section, we explain the multiscale high-dimensional sparse Fourier transform whose pseudocode is provided in Algorithm 4. The set $R$ contains the identified Fourier frequencies and their corresponding coefficients, and it is an empty set initially. Parameter $t$ is the counting number determining the index $\widetilde{k}$ of the coordinate axes where the frequencies are projected in line 6. Parameters $p, \tau$ and $L$ are determined as discussed in the previous sections. Function $Q$ in line 7 is a function constructed from the previously found Fourier modes which is used in updating our samples in lines 9 and 17. In line 9, the unshifted samples $\boldsymbol{r}_p^{\widetilde{k}}$ corrupted by random Gaussian noise $z_\ell$ are taken and in line 11, DFT is applied to these samples and the transformed vector is sorted in the descending order of magnitude. Only its $s^*$ largest components are taken into account under the $s^*$-sparsity assumption. In the loop from line 12 through 39, the entries of frequencies corresponding to these $s^*$ components are estimated iteratively. The shift size $\epsilon_q$ is updated in line 14 and we get the $D'$ number of length $p$ samples at the points shifted by $\epsilon_q$ along each axis. Each length $p$ sample will be used to approximate each entry. Similar to line 11, DFT is applied to each length-$p$ sample and the transformed vector is sorted again following the index order of the sorted DFT of unshifted samples in line 19. In line 22, we check whether the $D'$ tests are passed at the same time or not. If not, the *vote* is increased by 1. From lines 24 through 34, the estimate $n'_k$ for $k$th entry is updated. Except when $q = 0$, $n'_k$ is improved by adding the correction term shown in line 29 in each iteration. In the last iteration when $q = L$, $n'_k$ is rounded to the nearest integer in order to recover the exact entry, as guaranteed by Corollary 1. Whether this estimate is stored in the set $R$ or not is determined by checking if the *vote* after $L$ iterations is less than $\eta(L+1)$. If it is less, this implies that the failure rate of the collision detection tests is less than $\eta$. Accordingly, we estimate $c_{\boldsymbol{n}'}$ from the DFT of unshifted samples and store $(\boldsymbol{n}', c_{\boldsymbol{n}'})$ in $R$. The entire while loop repeats until $s$ energetic Fourier modes are all found through switching the projection coordinate. Once we find all $s$ frequency vectors, each $\boldsymbol{n}' \in R$ is transformed to $D$-dimensional $\boldsymbol{n} = u^{-1}(\boldsymbol{n}')$.

**Theorem 5.** *Let $f^z(\boldsymbol{x}) = f(\boldsymbol{x}) + z(\boldsymbol{x})$, where $\hat{f}(\boldsymbol{n})$ is $s$-sparse with all frequencies satisfying $\boldsymbol{n} \in \mathcal{S} \subset [-M/2, M/2)^D \cap \mathbb{Z}^D$ and not forming any worst case scenario, and $z$ is complex i.i.d. Gaussian noise of variance $\sigma^2$. Moreover, suppose that $s > C(\beta(\beta+1)c_{\min}\sigma)^2$ for some constant $C$. Algorithm 4, given $M, D, s, \beta$ with $M > 5s$ and access to $f^z(\boldsymbol{x})$ returns a list of $s$ pairs $(\hat{\boldsymbol{n}}, c_{\hat{\boldsymbol{n}}})$ such that (i) each $\hat{\boldsymbol{n}} \in \mathcal{S}$ and (ii) for each $\hat{\boldsymbol{n}}$, there is an $\boldsymbol{n} \in \mathcal{S}$ such that $|c_{\boldsymbol{n}} - c_{\hat{\boldsymbol{n}}}| \leq C\sigma/\sqrt{s}$. The average-case runtime and sampling complexity are*

$$\mathcal{O}(sD \log s \log M) \qquad \text{and} \qquad \mathcal{O}(sD \log M),$$

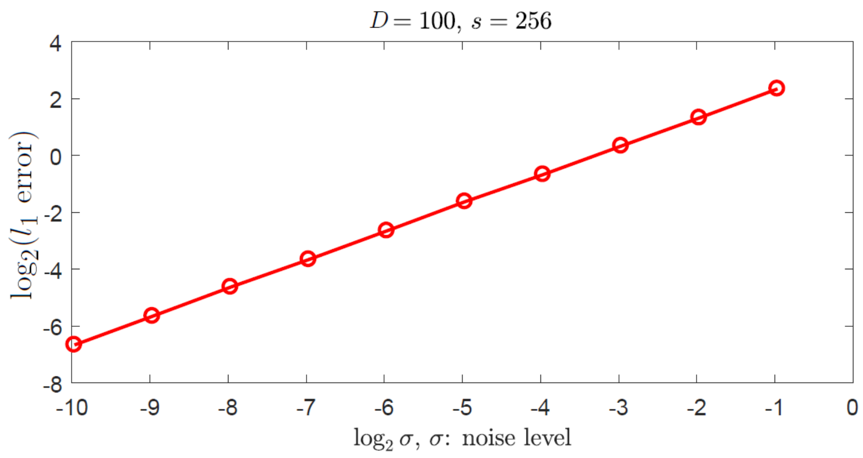
*respectively, over the class of random signals.*

*Proof.* The difference of Algorithm 4 from Algorithm 3 appears in lines 13 through 39. Algorithm 4 has the multiscale frequency entry estimation. Thus, the average-case runtime and sampling complexity of Algorithm 3 is increased by a factor of $L$ which is the number of the repetition in the multiscale frequency entry estimation. Corollary 1 ensures that the returned frequency vectors $\hat{\boldsymbol{n}}$ are correct, and the coefficient $c_{\hat{\boldsymbol{n}}}$ has the desired error bound from (3.10). □

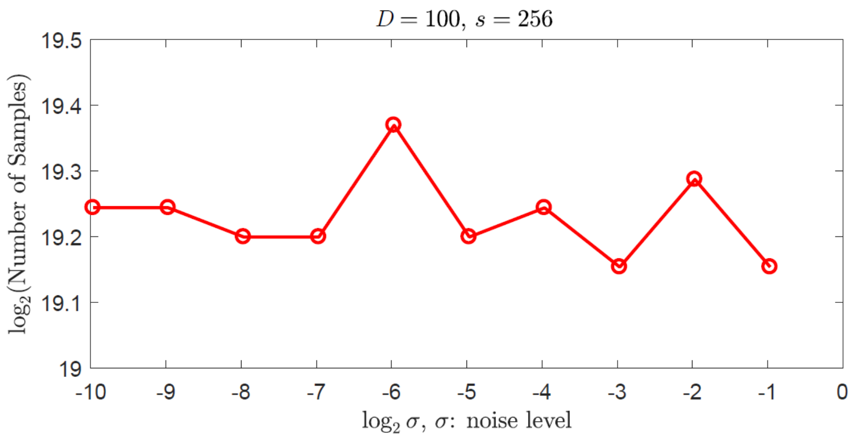
## 3.4 Empirical Evaluation

In this section, we show the empirical evaluation of the multiscale high-dimensional sparse Fourier algorithm. The empirical evaluation was done for test functions $f(\boldsymbol{x})$ which consist of $c_{\boldsymbol{n}}$ randomly chosen from a unit circle in $\mathbb{C}$ and $\boldsymbol{n}$ randomly chosen from $[-M/2, M/2)^D \cap \mathbb{Z}^D$. Sparsity $s$ varied from 1 to $2^{10} = 1024$ by factor of 2. The noise term added to each sample of $f$ came from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Standard deviation $\sigma$ varied from 0.001 to 0.512 by factor of 2. The dimension $D$ was chosen to be 100 and 1000, and $M$ was chosen as 20. The transformation $u$ was the one for partial unwrapping that was used in [35], i.e., every 5-dimensional subvector of each frequency vector was unwrapped to a one-dimensional vector and therefore each 100 and 1000-dimensional function $f$ was unwrapped to a 20 or 200-dimensional function $f \circ g$ whose Fourier domain is $\left([-20^5/2, 20^5/2) \cap \mathbb{Z}\right)^{20}$ or $\left([-20^5/2, 20^5/2) \cap \mathbb{Z}\right)^{200}$, respectively. Input parameters $C_1 = 2$, $C_\sigma = 6$, $\eta = 1/4$ and $\beta = 2.5$ were empirically chosen to balance the runtime and accuracy as in [2]. Initial shift size $\epsilon_0$ was set to $\frac{1}{2 \cdot 20^5}$. All experiments are performed in MATLAB.

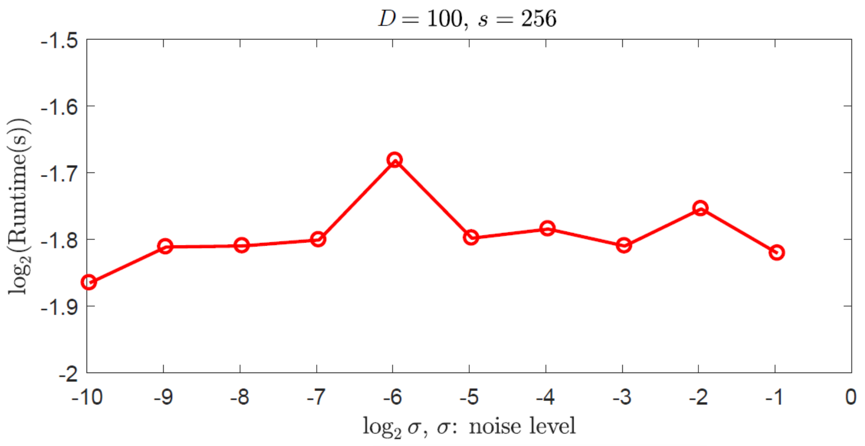
The three plots in Figure 3.1 show the average over 10 trials of the $l_1$ error, the number of samples, and the runtime in seconds as the noise level $\sigma$ changes. These values are in logarithm in the plots. Dimension $D$ and sparsity $s$ are fixed with 100 and 256, respectively. On the other hand, the other three plots in Figure 3.2 show the average over 10 trials of the $l_1$ error, the number of samples, and the runtime in seconds as the sparsity $s$ changes when $D = 100$ and 1000. These values are in logarithm in the plots, either and the noise level is fixed to 0.512.



(a)

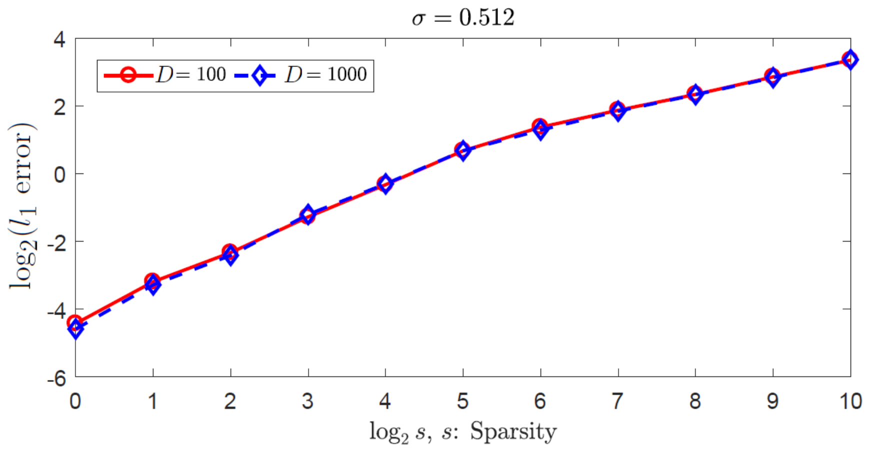


(b)

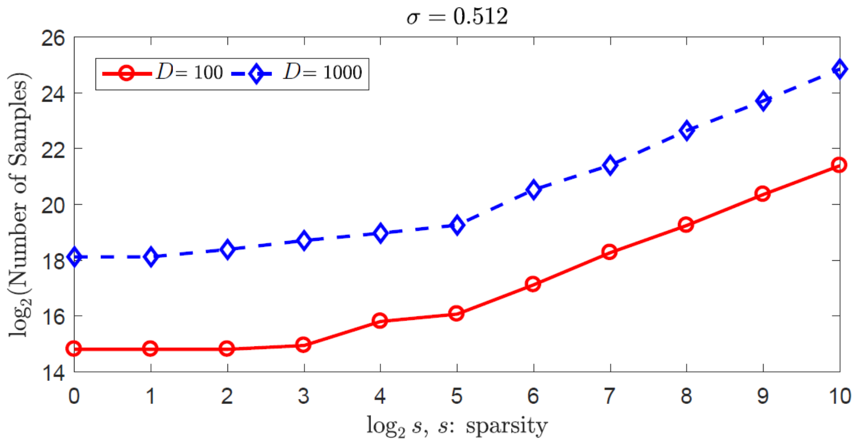


(c)

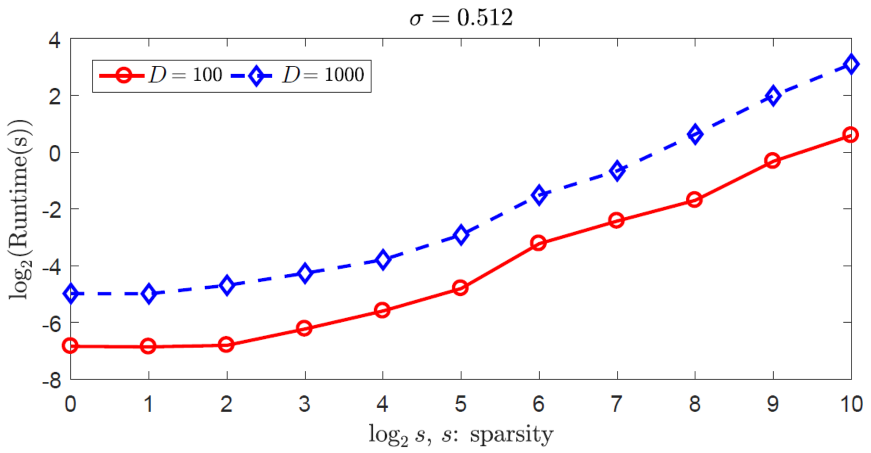
Figure 3.1: (a)Average $l_1$ error vs. noise level $\sigma$ in logarithm. (b)Average samples vs. noise level in logarithm. (c)Average runtime vs. noise level in logarithm.



(a)



(b)



(c)

Figure 3.2: (a)Average $l_1$ error vs. sparsity $s$ in logarithm. (b)Average samples vs. sparsity $s$ in logarithm. (c)Average runtime vs. sparsity $s$ in logarithm.

### 3.4.1 Accuracy

In Figure 3.1a and Figure 3.2a, the $l_1$ errors of Fourier coefficient vectors are given under various parameter changes. Throughout all trials conducted in these experiments frequencies were always recovered exactly even for the noise level of $\sigma = 0.512$, which is relatively large compared to the true coefficients from the unit circle in $\mathbb{C}$. Thus, we can observe the errors only from coefficients whose size is a constant multiple of $\frac{\sigma}{\sqrt{p}}$ from (3.10). Due to the characteristic of the *multiscale method* which uses less samples compared to the *rounding method*, $l_1$ errors are relatively large in nature. From Figure 3.1a, $l_1$ error looks increasingly linear as $\sigma$ increases, which meets our expectation. In Figure 3.2a, the plot does not look exactly linear, but between $\log_2 s = 5$ and 6, there is a transition of slope. This is because the sample length $p$ from (3.18) changes from $C_1 s$ to $\left(\frac{\beta(\beta+1)c_{\min}C_\sigma\sigma}{\pi}\right)^2$ during this transition.

### 3.4.2 Sampling complexity

Sample number along $\sigma$ changes in Figure 3.1b seems irregular at first sight. Looking at the scale of vertical axis, however, we can see that the difference between maximum and minimum is less than 0.3. Therefore, sampling complexity is not very affected by noise level. In Figure 3.2b, the red graph shows the average sample numbers as the sparsity increases when $D = 100$ and the blue graph shows the ones when $D = 1000$. Since our multiscale algorithm recovers each frequency entry iteratively using $\log M$ sets of $\mathcal{O}(s)$-length samples, the average-case sampling complexity is indeed $\mathcal{O}(sD\log M)$. Two graphs in Figure 3.2b look close to be linear excluding the transition between $\log_2 s = 5$ and 6, which again is caused by the change of $p$ from (3.18). Moreover the difference between the values of the red and blue graphs are close to 3, which implies that the sampling number depends linearly on $D$. The $D$-dimensional FFT whose sampling complexity is $\mathcal{O}(M^D)$ cannot deal with our high-dimensional problem computationally, whereas our algorithm uses only millions to billions of samples for reconstruction.

### 3.4.3 Runtime complexity

Figures 3.1c and 3.2c show the average-case runtime complexity of the algorithm. The time for evaluating the samples from functions is excluded when measuring the runtime. For the main algorithm, we demonstrated that it is $\mathcal{O}(sD \log s \log M)$ because for each entry recovery, DFT with $\mathcal{O}(s \log s)$ runtime complexity is applied in $D \log M$ iterations. In Figures 3.1c, the runtimes in seconds look irregular but the scale of vertical axis is less than 0.3 so that we can conclude that similar to sample numbers the runtime is not affected by $\sigma$ very much. Overall, it took less than a second on average. In Figure 3.2c, the red graph represents the runtimes as the sparsity changes when $D = 100$, and the blue graph represents the ones when $D = 1000$. Those graphs do not look linear, but considering the average slope we can see that the runtime is increased by around $2^8$ while $s$ is increased by $2^{10}$. On the other hand, the difference between two graphs implies that the runtime complexity is linear in $D$. Compared to the FFT with runtime complexity of $\mathcal{O}(M^D \log M^D)$ which is impossible to be practical in high-dimensional problems, our algorithm is quite effective, taking only a few seconds.

**Algorithm 4** Multiscale High-dimensional Sparse Fourier Algorithm Pseudo Code

```
    Input:$f, u, s, M, D, M', D', \sigma, c_{\min}, C_\sigma, C_1, \eta, \beta$
    Output:$R$
 1: $R \leftarrow \emptyset$, $t \leftarrow 0$
 2: while $|R| < s$ do
 3:     $s^* \leftarrow s - |R|$
 4:     $p \leftarrow$ first prime number $\geq \max\left\{C_1 s^*, \left(\beta(\beta+1)c_{\min}C_\sigma\sigma/\pi\right)^2\right\}$
 5:     $\tau \leftarrow \frac{C_\sigma \sigma}{c_{\min}\sqrt{p}}$, $L \leftarrow 1 + \lfloor \log_\beta M' \rfloor$
 6:     $\widetilde{k} \leftarrow (t \bmod D') + 1$
 7:     $Q(\boldsymbol{x}) \leftarrow \sum_{(\boldsymbol{n}', c_{\boldsymbol{n}'}) \in R} c_{\boldsymbol{n}'} e^{2\pi i \boldsymbol{n}' \cdot \boldsymbol{x}}$
 8:     for $\ell = 0 \rightarrow p-1$ do
 9:         $r_p^{\widetilde{k}}[\ell] \leftarrow f\left(u\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}}\right)\right) + z_\ell - Q\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}}\right)$
10:     end for
11:     $\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right) \leftarrow \mathrm{FFT}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)$, $\mathcal{F}^{\mathrm{sort}}\left(\boldsymbol{r}_p^{\widetilde{k}}\right) \leftarrow \mathrm{SORT}\left(\mathcal{F}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)\right)$
12:     $vote \leftarrow 0$
13:     for $q = 0 \rightarrow L$ do
14:         $\epsilon_q \leftarrow \frac{\beta^q}{2M'}$
15:         for $k = 1 \rightarrow D'$ do
16:             for $\ell = 0 \rightarrow p-1$ do
17:                 $r_{p,\epsilon_q}^{\widetilde{k},k}[\ell] \leftarrow f\left(u\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon_q \boldsymbol{e}_k\right)\right) + z'_\ell - Q\left(\frac{\ell}{p}\boldsymbol{e}_{\widetilde{k}} + \epsilon_q \boldsymbol{e}_k\right)$
18:             end for
19:             $\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right) \leftarrow \mathrm{FFT}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right)$, $\mathcal{F}^{\mathrm{sort}}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right) \leftarrow \mathrm{SORT}\left(\mathcal{F}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right)\right)$
20:         end for
21:         for $h = 0 \rightarrow s^* - 1$ do
22:             if $\left|\left|\mathcal{F}^{\mathrm{sort}}(\boldsymbol{r}_p^{\widetilde{k}})[h]\right| / \left|\mathcal{F}^{\mathrm{sort}}(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k})[h]\right| - 1\right| > \tau$ for any $k = 1, 2, \cdots, D'$ then $vote \leftarrow vote + 1$
23:             end if
24:             for $k = 1 \rightarrow D'$ do
25:                 $b_k \leftarrow \frac{1}{2\pi}\mathrm{Arg}\left(\left(\mathcal{F}^{\mathrm{sort}}\left(\boldsymbol{r}_{p,\epsilon_q}^{\widetilde{k},k}\right)[h]\right) / \left(\mathcal{F}^{\mathrm{sort}}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]\right)\right)$
26:                 if $q == 0$ then
27:                     $n'_k \leftarrow b_k / \epsilon_q$
28:                 else
29:                     $n'_k \leftarrow n'_k + \left(\left(b_k - \epsilon_q n'_k\right) (\mathrm{mod}\ [-1/2, 1/2))\right) / \epsilon_q$
30:                 end if
31:                 if $q == L$ then
32:                     $n'_k \leftarrow round\left(n'_k\right)$
33:                 end if
34:             end for
35:             if $vote \leq \eta(L+1)$ then
36:                 $c_{\boldsymbol{n}'} \leftarrow \frac{1}{p}\mathcal{F}^{\mathrm{sort}}\left(\boldsymbol{r}_p^{\widetilde{k}}\right)[h]$, $R \leftarrow R \cup \left(\boldsymbol{n}', c_{\boldsymbol{n}'}\right)$
37:             end if
38:         end for
39:     end for
40:     $t \leftarrow t + 1$
41: end while
42: inverse-transform each $\boldsymbol{n}'$ in $D'$-D to $\boldsymbol{n}$ $D$-D and restore it in $R$
```

# Chapter 4

# General Sparse Harmonic Transforms

## 4.1 Preliminaries

In this section, we introduce a completely different approach for high dimensional sparse problems. The approach is a generalized sparse transform for a generic tensor basis. We present the new method and a class of problems that it is ideally suited for.

### 4.1.1 The Compressive Sensing Problem for BOPB-Sparse Functions

Toward a more exact problem formulation, let $p \in \mathbb{N}$ be any natural number and $[N] := \{0, 1, 2, \ldots, N-1\}$ for all $N \in \mathbb{N}$. The set of functions, $\{T_k : \mathcal{D} \to \mathbb{C}\}_{k \in [N]}$ forms a *Bounded Orthonormal System* (BOS) with respect to a probability measure $\sigma$ over $\mathcal{D} \subset \mathbb{R}^p$ with BOS constant $K := \max_k \|T_k\|_\infty \geq 1$ if $K < \infty$, and

$$\langle T_k, T_l \rangle_{(\mathcal{D},\sigma)} := \int_{\mathcal{D}} T_k(\boldsymbol{x}) \overline{T_l(\boldsymbol{x})} d\sigma(\boldsymbol{x}) = \delta_{k,l} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases}$$

holds for all $k, l \in [N]$. Now let $\mathcal{B}_j := \{T_{j,k} : \mathcal{D}_j \to \mathbb{C}\}_{k\in[M]}$ form a BOS with respect to a probability measure $\nu_j$ on $\mathcal{D}_j \subset \mathbb{R}$, with constant $\widetilde{K}_j$ for each $j \in [D]$. Then, the BOPB functions $\mathcal{B} := \{T_{\boldsymbol{n}} : \mathcal{D} \to \mathbb{C}\}_{\boldsymbol{n}\in[M]^D}$, defined by

$$T_{\boldsymbol{n}}(\boldsymbol{x}) := \prod_{j\in[D]} T_{j;n_j}(x_j) \tag{4.1}$$

again form a BOS with constant

$$K := \max_{\boldsymbol{n}} ||T_{\boldsymbol{n}}||_\infty = \prod_{j\in[D]} \widetilde{K}_j$$

with respect to the probability measure $\boldsymbol{\nu} := \otimes_{j\in[D]}\nu_j$ over $\mathcal{D} := \times_{j\in[D]}\mathcal{D}_j \subset \mathbb{R}^D$.

Herein we consider BOPB-sparse functions $f : \mathcal{D} \to \mathbb{C}$ of the form

$$f(\boldsymbol{x}) := \sum_{\boldsymbol{n}\in\mathcal{S}\subset\mathcal{I}\subseteq[M]^D} c_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x})$$

where $|\mathcal{S}| = s \ll |\mathcal{I}| \leq |\mathcal{B}| = N = M^D$. Following [32, 20, 18] we will take $\mathcal{I}$ to be the subset of $[M]^D$ containing at most $d \leq D$ nonzero entries.

Considering the recovery of $f$ using standard compressive sensing methods [36, 37] when $\mathcal{I} = [M]^D$, one can simply independently draw $m_1'$ points, $\mathcal{G}^E := \left\{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{m_1'}\right\} \subset \mathcal{D}$, according to $\boldsymbol{\nu}$ and then sample $f$ at those points to obtain

$$\boldsymbol{y}^{\mathbf{E}} = f\left(\mathcal{G}^E\right) := \left(f(\boldsymbol{t}_1), f(\boldsymbol{t}_2), \ldots, f\left(\boldsymbol{t}_{m_1'}\right)\right)^T \in \mathbb{C}^{m_1'}. \tag{4.2}$$

Our objective becomes the recovery of $f$ using only the samples $\boldsymbol{y}^{\mathbf{E}}$.

Let the $m_1' \times M^D$ *random sampling matrix* $\Phi \in \mathbb{C}^{m_1' \times M^D}$ have entries given by

$$\Phi_{\ell,\boldsymbol{n}} = T_{\boldsymbol{n}}(\boldsymbol{t}_\ell). \tag{4.3}$$

We can now form the underdetermined linear system

$$\boldsymbol{y}^{\mathbf{E}} = \begin{pmatrix} f(\boldsymbol{t}_1) \\ f(\boldsymbol{t}_2) \\ \vdots \\ f\left(\boldsymbol{t}_{m_1'}\right) \end{pmatrix} = \begin{pmatrix} T_{\boldsymbol{n}_1}(\boldsymbol{t}_1) & T_{\boldsymbol{n}_2}(\boldsymbol{t}_1) & \cdots & \cdots & T_{\boldsymbol{n}_{M^D}}(\boldsymbol{t}_1) \\ T_{\boldsymbol{n}_1}(\boldsymbol{t}_2) & T_{\boldsymbol{n}_2}(\boldsymbol{t}_2) & \cdots & \cdots & T_{\boldsymbol{n}_{M^D}}(\boldsymbol{t}_2) \\ \vdots & \vdots & & \ddots & \vdots \\ T_{\boldsymbol{n}_1}(\boldsymbol{t}_{m_1'}) & T_{\boldsymbol{n}_2}(\boldsymbol{t}_{m_1'}) & \cdots & \cdots & T_{\boldsymbol{n}_{M^D}}(\boldsymbol{t}_{m_1'}) \end{pmatrix} \boldsymbol{c} = \Phi \boldsymbol{c},$$

where $\boldsymbol{c} \in \mathbb{C}^{M^D}$ contains the basis coefficients $c_{\boldsymbol{n}}$ of $f$, and the index vectors $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_{M^D} \in [M]^D$ are ordered, e.g., lexicographically. Note that this linear system is woefully underdetermined when $m_1' \ll M^D$. When $\boldsymbol{c}$ has only $s \ll M^D$ nonzero entries as it does here, however, the compressive sensing literature tells us that $\boldsymbol{c}$ can still be recovered using significantly fewer than $M^D$ function evaluations as long as the normalized random sampling matrix $\Phi$ has the *Restricted Isometry Property* (RIP) of order $2s$ [37].

**Definition 1** (See Definition 6.1 in [37])**.** *The $s^{\text{th}}$ restricted isometry constant $\delta_s$ of a matrix $\widetilde{\Phi} \in \mathbb{C}^{m \times N}$ is the smallest $\delta \geq 0$ such that*

$$(1-\delta)\|\boldsymbol{c}\|_2^2 \leq \left\|\widetilde{\Phi}\boldsymbol{c}\right\|_2^2 \leq (1+\delta)\|\boldsymbol{c}\|_2^2$$

*holds for all $s$-sparse vectors $\boldsymbol{c} \in \mathbb{C}^N$. The matrix $\widetilde{\Phi}$ is said to satisfy the RIP of order $s$ if $\delta_s \in (0,1)$.*

Furthermore, one can show that random sampling matrices have the restricted isometry property with high probability when $\left|\mathcal{G}^E\right|$ is relatively small.

**Theorem 6** (See Theorem 12.32 and Remark 12.33 in [37])**.** *Let $A \in \mathbb{C}^{m\times N}$ be the random sampling matrix associated to a BOS with constant $K \geq 1$. If, for $\delta, p \in (0,1)$,*

$$m \geq aK^2\delta^{-2}s \cdot \max\{\log^2(4s)\log(8N)\log(9m), \log(p^{-1})\},$$

*then with probability at least $1-p$, the restricted isometry constant $\delta_s$ of $\widetilde{A} = \frac{1}{\sqrt{m}}A$ satisfies $\delta_s \leq \delta$. The constant $a > 0$ is universal.*

Note that Theorem 6 effectively decouples the number of samples that one must acquire/compute in order to recover any BOPB-sparse $f$ from the overall BOP basis size $|\mathcal{B}| = M^D$. It guarantees that a random sampling set of size $\left|\mathcal{G}^E\right| = m_1' = \mathcal{O}(K^2 \cdot s \cdot D \cdot \log^4(KMD))$ suffices. The main obstacle to reducing the sampling complexity (i.e., $m_1'$) at this point becomes the BOS sampling constant $K$. To see why, consider, e.g., the cosine BOPB where for all $j \in [D]$ in (4.1) we set $T_{j;n}(x) = \sqrt{2}\cos(nx)$ for $n \geq 1$ and $T_{j;0}(x) = 1$ in (4.1). This leads to a BOS with $K = 2^{D/2}$ with respect to uniform probability measure $\boldsymbol{\nu}$ over $\mathcal{D} = [0, 2\pi]^D$. Now we can see that we still face the curse of dimensionality since $K^2 = 2^D$ even for this fairly straightforward BOPB. Nonetheless, it expresses itself in a dramatically reduced fashion: $2^D$ is still a vast improvement over $M^D$ for even moderately sized $M > 2$.

As previously mentioned, to further reduce the sampling complexity from scaling like $2^{\mathcal{O}(D)}$ previous work has focussed on developing efficient methods for effectively reducing the basis size to a smaller subset of the total basis $\mathcal{B}$ (see, e.g., [19, 20]). To see how this might work in the context of our simple cosine BOPB example above, we can note that the BOPB elements in (4.1) can be rewritten as

$$T_{\boldsymbol{n}}(\boldsymbol{x}) := 2^{\|\boldsymbol{n}\|_0/2} \prod_{j=0}^{D-1} \cos\left(n_j x_j\right)$$

in that case. It now becomes obvious that limiting the basis functions to those with indexes in $\mathcal{I} := \{\boldsymbol{n} \in [M]^D \mid \|\boldsymbol{n}\|_0 \leq d \leq D\}$ leads to a reduced BOS constant of $K = 2^{d/2} < 2^{D/2}$ for the resulting reduced basis, as well as to a smaller basis cardinality of size $\binom{D}{d} M^d = \mathcal{O}\left((\frac{DM}{d})^d\right)$.

In particular, the utility of the assumption that the $s$ non-negligible basis indexes of $f$, $\mathcal{S} \subset [M]^D$, also belong to the reduced index set $\mathcal{I}$ above is supported in some UQ applications where it is known that, e.g., the solutions of some parametric PDE are not only approximately sparse in some BOP bases such as the Chebyshev or Legendre product bases, but also that most of their significant coefficients correspond to index vectors $\boldsymbol{n} \subset \mathbb{N}^D$ with relatively small (weighted) $\ell_p$-norms [19, 20]. In certain simplified situations this essentially implies that $\mathcal{S} \subset \mathcal{I}$ as discussed above. As a result, we will assume throughout this chapter that $\mathcal{S} \subset \mathcal{I}$ so that $N = |\mathcal{I}| = \binom{D}{d} M^d \leq M^D$.[1]

Even when $s{\cdot}K^2 \ll N$ so that the number of required samples $m_1'$ is small compared to the reduced basis size $|\mathcal{I}| = N$, however, all existing standard compressive sensing approaches for recovering $f$ still need to compute and store potentially fully populated intermediate coefficient vectors $\boldsymbol{c}' \in \mathbb{C}^N$ at some point in the process of recovering $f$. As a result, all existing approaches are limited in terms of the reduced basis sizes $\mathcal{I}$ they can consider

[1]Additionally, we will occasionally assume that our total grid size $|\mathcal{G}|$ below always satisfies $|\mathcal{G}| \leq N^c$ for some absolute constant $c \geq 1$ in order to simplify some of the logarithmic factors appearing in our big-O notation. This will certainly always be the case for any standardly used (trigonometric) polynomial BOPB (such as Fourier and Chebyshev product bases) whenever $sKDM < N$.

by both their memory needs and runtime complexities. In this chapter we develop new methods that are capable of circumventing these memory and runtime restrictions for a general class of practical BOP bases. As a result, we make it possible to recover a new class of extremely high-dimensional BOPB-sparse functions which are simply too complicated to be approximated by other means. We are now prepared to discuss our main results.

### 4.1.2 Main Results

The proposed sublinear-time algorithm is a greedy pursuit method motivated by CoSaMP[22], HTP[23], and their sublinear-time predecessors [38, 39]. In particular, it is obtained from CoSaMP by replacing CoSaMP's support identification procedure with a new sublinear-time support identification procedure. See Algorithm 5 in section 4.2 for pseudocode and other details. Our main result demonstrates the existence of a relatively small grid of points $\mathcal{G} \subset \mathcal{D}$ which allows Algorithm 5 to recover any given BOPB-sparse function $f$ in sublinear-time from its evaluations on $\mathcal{G}$. We refer the reader to section 4.1.3 for a detailed description of the grid set $\mathcal{G}$ and its use in Algorithm 5. The following theorem is a simplified version of theorem 7 in section 4.2.

**Theorem** (Main Result)**.** *Suppose that* $\left\{T_{\boldsymbol{n}} \;\middle|\; \boldsymbol{n} \in \mathcal{I} \subseteq [M]^D\right\}$ *is a BOS where each basis function* $T_{\boldsymbol{n}}$ *is defined as per* (4.1)*. Let* $\mathcal{F}_s$ *be the subset of all functions* $f \in \operatorname{span}\left\{T_{\boldsymbol{n}} \;\middle|\; \boldsymbol{n} \in \mathcal{I}\right\}$ *whose coefficient vectors are s-sparse, and let* $\boldsymbol{c}_f \in \mathbb{C}^{\mathcal{I}}$ *denote the s-sparse coefficient vector for each* $f \in \mathcal{F}_s$*. Fix* $p \in (0, 1/3)$*, a precision parameter* $\eta > 0$*,* $1 \leq d \leq D$*, and* $K = \sup_{\boldsymbol{n} \text{ s.t.} \|\boldsymbol{n}\|_0 \leq d} \|T_{\boldsymbol{n}}\|_\infty$*. Then, one can randomly select a set of i.i.d. Gaussian weights* $\mathcal{W} \subset \mathbb{R}$ *for use in* (4.19)*, and also randomly construct a compressive sensing grid,* $\mathcal{G} \subset \mathcal{D}$*, whose total cardinality* $|\mathcal{G}|$ *is* $\mathcal{O}\left(s^3 D \mathcal{L}' K^4 \max\left\{d^4 \log^4(s) \log^4(D^2 M), \log^2(\frac{D}{p})\right\}\right)$*, such that* *the following property holds $\forall f \in \mathcal{F}_s$ with probability greater than $1-3p$: Let $\boldsymbol{y} = f(\mathcal{G})$ consist of samples from $f \in \mathcal{F}_s$ on $\mathcal{G}$. If Algorithm 5 is granted access to $\boldsymbol{y}$, $\mathcal{G}$, and $\mathcal{W}$, then it will produce an $s$-sparse approximation $\boldsymbol{a} \in \mathbb{C}^{\mathcal{I}}$ s.t.*

$$\|\boldsymbol{c}_f - \boldsymbol{a}\|_2 \leq C\eta,$$

*where $C > 0$ is an absolute constant. Furthermore, the total runtime complexity of Algorithm 5 is always*
$\mathcal{O}\Big(\Big(s^5 D^2 \mathcal{L} K^4 \max\Big\{d^4 \log^4(s)\log^4(D^2 M), \log^2(\frac{D}{p})\Big\}\Big) \times \log\frac{\|\boldsymbol{c}_f\|_2}{\eta}\Big).$

Note that Algorithm 5 will run in sublinear-time whenever $s^5 D\mathcal{L}K^4 d^4 \ll |\mathcal{I}|$ (neglecting logarithmic factors). Here and in the theorem above the parameters $\mathcal{L}$ and $\mathcal{L}'$ depend on your choice of numerical method for computing the inner product between a sparse function in the span of each one-dimensional BOS $\mathcal{B}_j = \{T_{j;m} \mid m \in [M]\}$. More specifically, let $\mathcal{L}'_j$ represent the number of function evaluations one needs in order to compute all $M$-inner products $\Big\{\langle g, T_{j;\widetilde{n}}\rangle\Big\}_{\widetilde{n}\in[M]}$ in $\mathcal{O}(\mathcal{L})$-time for any given function $g : \mathcal{D}_j \to \mathbb{C}$ belonging to the span of $\mathcal{B}_j$ that is also $s$-sparse in $\mathcal{B}_j$. We then set $\mathcal{L}' := \max_{j\in[D]} \mathcal{L}'_j$. For example, if each BOS $\mathcal{B}_j$ consists of orthonormal polynomials whose degrees are all bounded above by $M$ then quadrature rules such as Gaussian quadrature or Chebyshev quadrature give $\mathcal{L} = \mathcal{O}(M^2)$ and $\mathcal{L}' = \mathcal{O}(M)$ [14]. If each $\mathcal{B}_j$ is either the standard Fourier, sine, cosine, or Chebyschev basis then the Fast Fourier Transform (FFT) can always be used to give $\mathcal{L} = \mathcal{O}(M \log M)$ and $\mathcal{L}' = \mathcal{O}(M)$ [14].

Moreover, there are several sublinear-time sparse Fourier transforms as well as sparse harmonic transforms for other bases which could also be used to give other valid $\mathcal{L}'$ and $\mathcal{L}$ combinations [40, 41, 10, 5, 24, 42, 43, 7, 44, 8, 45, 11, 46]. These typically have $\mathcal{O}(s^c \log^{c'} M)$

runtime and sampling complexities for small positive absolute constants $c$ and $c'$. As a result, one can obtain much stronger results than the main theorem above when $s \ll M$ and every one-dimensional BOS $\mathcal{B}_j$ is either the Fourier, sine, cosine, or Chebyshev basis. The following corollary of our main theorem is obtained by using deterministic one-dimensional SFT results from [11] and [44] in order to compute all of the nonzero inner products in lines 6 – 13 of Algorithm 6. They lead to $\mathcal{L}'$ and $\mathcal{L}$ values in section 4.2's theorem 7 of size $\mathcal{O}(s^2 \log^4 M)$.

**Corollary 2.** *Suppose that* $\left\{T_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathcal{I} \subseteq [M]^D\right\}$ *is a BOS where each basis function* $T_{\boldsymbol{n}}$ *is defined as per* (4.1)*, and where every one-dimensional BOS* $\mathcal{B}_j$ *is either the Fourier, sine, cosine, or Chebyshev basis. Let* $\mathcal{F}_s$ *be the subset of all functions* $f \in \operatorname{span}\left\{T_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathcal{I}\right\}$ *whose coefficient vectors are s-sparse, and let* $\boldsymbol{c}_f \in \mathbb{C}^{\mathcal{I}}$ *denote the s-sparse coefficient vector for each* $f \in \mathcal{F}_s$*. Fix* $p \in (0, 1/3)$*, a precision parameter* $\eta > 0$*,* $1 \leq d \leq D$*, and let* $K = \sup_{\boldsymbol{n} \text{ s.t.} \|\boldsymbol{n}\|_0 \leq d} \|T_{\boldsymbol{n}}\|_\infty$*. Then, one can randomly select a set of i.i.d. Gaussian weights* $\mathcal{W} \subset \mathbb{R}$ *for use in* (4.19)*, and also randomly construct a compressive sensing grid,* $\mathcal{G} \subset \mathcal{D}$*, whose total cardinality* $|\mathcal{G}|$ *is*
$\mathcal{O}\left(s^3 D \log^4(M) K^4 \max\left\{d^4 \log^4(s) \log^4(D^2 M), \log^2(\frac{D}{p})\right\}\right)$*, such that the following property holds* $\forall f \in \mathcal{F}_s$ *with probability greater than* $1 - 3p$*: Let* $\boldsymbol{y} = f(\mathcal{G})$ *consist of samples from* $f \in \mathcal{F}_s$ *on* $\mathcal{G}$*. If Algorithm 5 is granted access to* $\boldsymbol{y}$*,* $\mathcal{G}$*, and* $\mathcal{W}$*, then it will produce an s-sparse approximation* $\boldsymbol{a} \in \mathbb{C}^{\mathcal{I}}$ *s.t.*

$$\|\boldsymbol{c}_f - \boldsymbol{a}\|_2 \leq C\eta,$$

*where* $C > 0$ *is an absolute constant. Furthermore, the total runtime complexity of Algorithm 5 is always*

$$\mathcal{O}\Big(\Big(s^5 D^2 \log^4(M) K^4 \max\Big\{d^4 \log^4(s)\log^4(D^2M), \log^2(\tfrac{D}{p})\Big\}\Big) \times \log\frac{\|\boldsymbol{c}_f\|_2}{\eta}\Big).$$

Note that the runtime dependance achieved by the corollary above scales sublinearly with $M$, quadratically in $D$, and at most polynomially in the parameter $d \leq D$ used to determine $\mathcal{I}$. We also remind the reader that the BOS constant $K$ for the Fourier basis is 1. As a result, the $K$ dependence in the runtime complexity vanishes entirely when the BOPB in question is the multidimensional Fourier basis.[2] Finally, there are also sublinear-time sparse transforms for one-dimensional Legendre polynomial systems [44], though the theoretical results for sparse recovery therein require additional support restrictions beyond simple sparsity. Thus, corollary 2 can also be extended to restricted types of Legendre-sparse functions in order to achieve sublinear-in-$M$ runtimes. A detailed development of such results is left for future work, however.

### 4.1.3 Randomly Constructed Grids with Universal Approximation Properties

Fix a BOP basis $\mathcal{B}$ and sparsity level $s$. We will call any set $\mathcal{G} \subset \mathcal{D}$ a *compressive sensing grid* if and only if $\exists$ a set of weights $\mathcal{W}$ s.t. $\forall\ f : \mathcal{D} \to \mathbb{C}$ that are $s$-sparse in $\mathcal{B}$

Algorithm 5 with weights $\mathcal{W}$ can recover $f$ from its evaluations on $\mathcal{G}$

is true. As mentioned above, our main results demonstrate the existence of relatively small compressive sensing grids by randomly constructing highly structured sets of points that are

[2]Though the resulting $\mathcal{O}\left(s^5D^2d^4\text{polylog}(MDs\|\boldsymbol{c}\|_2/\eta p)\right)$-runtime achieved by corollary 2 for the multidimensional Fourier basis is strictly worse than the best existing noise robust and deterministic sublinear-time results for that basis [11] (except perhaps when $s^3d^4 \ll D^3$), we emphasize that it is achieved with a different and significantly less specialized grid $\mathcal{G}$ herein.

then shown to be compressive sensing grids with high probability. We emphasize that our use of probability in this chapter is entirely constrained to $(i)$ the initial choice of the grid $\mathcal{G}$ given a BOP basis $\mathcal{B}$ and sparsity level $s$, and to $(ii)$ the entirely independent and one-time initial choice of a set of random gaussian weights $\mathcal{W}$ for use in (4.19) (i.e., as part of the initialization phase for Algorithm 5). Algorithm 5 is entirely deterministic once both $\mathcal{G}$ and $\mathcal{W}$ have been chosen.

The compressing sensing grids $\mathcal{G}$ utilized herein will be the union of three distinct sets of points in $\mathcal{D}$. The first set of points is the set $\mathcal{G}^E \subset \mathcal{D}$ on which $f$ is evaluated in order to obtain $\boldsymbol{y}^{\mathbf{E}}$ in (4.2). This set of points is used in Algorithm 5 in order to estimate the basis coefficients for the basis elements identified by Algorithms 6 and 7. The second and third sets included in $\mathcal{G}$, $\mathcal{G}^I \subset \mathcal{D}$ and $\mathcal{G}^P \subset \mathcal{D}$, are utilized by Algorithm 6 and Algorithm 7, respectively. They are defined below.

Let $\mathcal{U}_j := \left\{u_{j,0}, \dots, u_{j,\mathcal{L}'_j-1}\right\} \subset \mathcal{D}_j$ be the set of $\mathcal{L}'_j$ points at which one can evaluate any given $\mathcal{B}_j$-sparse function $g : \mathcal{D}_j \to \mathbb{C}$ in the span of $\mathcal{B}_j$ in order to compute all $M$-inner products $\left\{\langle g, T_{j;\widetilde{n}}\rangle\right\}_{\widetilde{n}\in[M]}$ in $\mathcal{O}(\mathcal{L})$-time. Also, let $I_{\geq j} : [D-1] \to \{0,1\}$ be the indicator function that is zero when $\kappa < j$, and one when $\kappa \geq j$. For each $j \in [D]$ we will then define $\mathcal{G}^I_j \subset \mathcal{D}$ to be the set of $m\mathcal{L}'_j$ randomly generated grid points given by

$$\boldsymbol{x}'_{j,\ell,k} = \big((x_{j,\ell})_0, (x_{j,\ell})_1, \dots, (x_{j,\ell})_{j-1}, u_{j,k}, (x_{j,\ell})_j, \dots, (x_{j,\ell})_{D-2}\big) \;\; \forall(\ell,k) \in [m] \times [\mathcal{L}'_j],$$

where each $(x_{j,\ell})_\kappa \in \mathcal{D}_{\kappa+I_{\geq j}(\kappa)}$ is an independent realization of a random variable $\sim \nu_{\kappa+I_{\geq j}(\kappa)}$ for all $\kappa \in [D-1]$. We now take $\mathcal{G}^I$ to be the union of these sets so that

$$\mathcal{G}^I := \bigcup_{j\in[D]} \mathcal{G}^I_j = \bigcup_{j\in[D]} \left\{\boldsymbol{x}'_{j,\ell,k}\right\}_{(\ell,k)\in[m]\times[\mathcal{L}'_j]}.$$

Finally, similar to (4.2), we will also define $f$'s evaluations on $\mathcal{G}^I$ to be $\boldsymbol{y}^{\mathbf{I}} \in \mathbb{C}^{m_2'}$ where

$$\boldsymbol{y}^{\mathbf{I}} = f\left(\mathcal{G}^I\right) := \left(f\left(\boldsymbol{x}'_{0,0,0}\right), f\left(\boldsymbol{x}'_{0,0,1}\right), \ldots, f\left(\boldsymbol{x}'_{D-1,m-1,\mathcal{L}'_{D-1}-1}\right)\right)^T.$$

To define $\mathcal{G}^P$ we will again need several different subsets for each $j \in [D] \setminus \{0\}$. For each fixed $(j, \ell, k) \in [D] \setminus \{0\} \times [m_1] \times [m_2]$ let $\boldsymbol{w}_{j,\ell} \in \times_{i \in [j+1]} \mathcal{D}_i$ and $\boldsymbol{z}_{j,k} \in \times_{i=j+1}^{D-1} \mathcal{D}_i$ be chosen independently at random according to $\otimes_{i \in [j+1]} \nu_i$ and $\otimes_{i=j+1}^{D-1} \nu_i$, respectively.[3] We then define $\mathcal{G}_j^P \subset \mathcal{D}$ to be the set of $m_1 m_2$ randomly generated grid points given by

$$\mathcal{G}_j^P := \left\{(\boldsymbol{w}_{j,\ell}, \boldsymbol{z}_{j,k}) \;\middle|\; (\ell, k) \in [m_1] \times [m_2]\right\} \;\; \forall j \in [D] \setminus \{0\}.$$

As above, we now let $\mathcal{G}^P$ be the union of these sets so that

$$\mathcal{G}^P := \bigcup_{j \in [D] \setminus \{0\}} \mathcal{G}_j^P$$

and define $f$'s evaluations on $\mathcal{G}^P$ to be $\boldsymbol{y}^{\mathbf{P}} \in \mathbb{C}^{m_3'}$ where

$$\boldsymbol{y}^{\mathbf{P}} = f\left(\mathcal{G}^P\right) := \left(f\left(\boldsymbol{w}_{1,0}, \boldsymbol{z}_{1,0}\right), f\left(\boldsymbol{w}_{1,0}, \boldsymbol{z}_{1,1}\right), \ldots, f\left(\boldsymbol{w}_{D-1,m_1-1}, \boldsymbol{z}_{D-1,m_2-1}\right)\right)^T.$$

As we saw in Section 4.1.2, it turns out that $\mathcal{G} := \mathcal{G}^E \cup \mathcal{G}^I \cup \mathcal{G}^P$ will be a compressive sensing grid with high probability even when each component set is chosen to have a relatively small cardinality. The vast majority of the remainder of this chapter will be dedicated to proving this fact. We will begin the remainder of Section 4.1 by introducing additional notation that is used throughout the rest of the chapter, and by interpreting our function

[3]When $j = D - 1$ the vector $\boldsymbol{z}_{j,k}$ is interpreted as a null vector satisfying $(\boldsymbol{w}_{j,\ell}, \boldsymbol{z}_{j,k}) = \boldsymbol{w}_{j,\ell}$ $\forall (\ell, k)$.

evaluations on $\mathcal{G}$,

$$\boldsymbol{y} = (\boldsymbol{y}^{\mathbf{E}}, \boldsymbol{y}^{\mathbf{I}}, \boldsymbol{y}^{\mathbf{P}})^T \in \mathbb{C}^{m_1' + m_2' + m_3'} \tag{4.4}$$

as standard compressive sensing measurements. Next, in Section 4.2, Algorithm 5 is discussed in detail and the main theorem above is proven with the help of a key technical lemma (i.e., Lemma 2) that guarantees the accuracy of our proposed support identification method. Lemma 2 is then proven in Section 4.3. Finally, a numerical evaluation is carried out in Section 4.4 that demonstrates that Algorithm 5 both behaves as expected, and is robust to noisy function evaluations.

### 4.1.4 Notation and Problem Setting

In this section we introduce the notation that will be used in the rest of this chapter as well as the problem for which we will develop our proposed algorithm. We denote by $\mathbb{N}$ the set of natural numbers, $\mathbb{R}$ the set of real numbers, and $\mathbb{C}$ the set of complex numbers. Let $[N] := \{0, 1, 2, \ldots, N-1\}$ for $N \in \mathbb{N}$.

In this chapter all letters in boldface (other than probability measures such as $\boldsymbol{\nu}$) will always represent vectors. Vectors whose entries are indexed by *index vectors* in, e.g., $[M]^D$ will be assumed to have their entries ordered lexicographically for the purposes of, e.g., matrix-vector multiplications. Thus, we say either $\boldsymbol{v} \in \mathbb{C}^{[M]^D}$ or $\boldsymbol{v} \in \mathbb{C}^{M^D}$ when we want to emphasize that each entry $v_{\boldsymbol{n}}$ of $\boldsymbol{v}$ is corresponding to its index vector $\boldsymbol{n}$, or when we perform, e.g., matrix-vector multiplications, respectively. We further define the $\ell_0$ pseudo-norm of a vector $\boldsymbol{v}$ by $\|\boldsymbol{v}\|_0 := |\{i : v_i \neq 0\}|$ where the index $i$ refers to the $i^{\text{th}}$ entry of the vector (in lexicographical order). If $v \in \mathbb{C}$ is a scalar then we will also use the $\ell_0$-notation to correspond to the indicator function defined by

$$\|v\|_0 := \begin{cases} 1 & \text{if } v \neq 0 \\ & \\ 0 & \text{if } v = 0 \end{cases}. \tag{4.5}$$

We will consider functions $f : \mathcal{D} \to \mathbb{C}$ given in a BOS product basis expansion below so that

$$f(\boldsymbol{x}) := \sum_{\boldsymbol{n} \in [M]^D} c_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x}). \tag{4.6}$$

We will further assume that $f$ is approximately sparse in this BOS product basis. That is, we will assume that there exists some index set $\mathcal{S} \subset \mathcal{I}$ for an a priori known index set $\mathcal{I} \subseteq [M]^D$ such that $\mathcal{S}$ has the property that both $(i)$ $|\mathcal{S}| = s \ll |\mathcal{I}| \leq M^D$, and that $(ii)$ the set of coefficients $\mathcal{C} := \{c_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathcal{S}\} \subset \mathbb{C}$ dominates $f$'s $\ell_2$-norm in the sense that

$$\sum_{\boldsymbol{n} \in \mathcal{S}} |c_{\boldsymbol{n}}|^2 \gg \sum_{\boldsymbol{n} \in [M]^D \setminus \mathcal{S}} |c_{\boldsymbol{n}}|^2 =: \epsilon^2,$$

for a relatively small number $\epsilon > 0$. We emphasize here that absolutely nothing about $\mathcal{S}$ is known to us in advance beyond the fact that it is a subset of $\mathcal{I}$, and has cardinality at most $s$. We must learn the identity of its elements ourselves by sampling $f$.

Our analysis herein will focus on the case where $\mathcal{I}$ is given by

$$\mathcal{I} := \left\{ \boldsymbol{n} \in [M]^D \mid \|\boldsymbol{n}\|_0 \leq d \right\}$$

for some $d \leq D$ (cf. Section 4.1.1). Note that this includes, for $d = D$, the special case where $\mathcal{I} = [M]^D$. We will call the index vectors $\boldsymbol{n} \in \mathcal{S}$ *energetic*. Our goal is to recover

$\mathcal{S}$ and the associated coefficients $\mathcal{C}$ as rapidly as possible using only evaluations/samples from $f$. This will, in turn, necessitate that we sample $f$ at very few locations in $\mathcal{D}$. In this chapter we will mainly focus on providing theoretical guarantees for the case where $\epsilon = 0$ (i.e., for provably recovering $f$ that are exactly $s$-sparse in a BOS product basis). Numerical experiments in Section 4.4 demonstrate that the method also works when $\epsilon > 0$, however. We leave theoretical guarantees in the case of $\epsilon > 0$ for future consideration.

### 4.1.5 Definitions Required for Support Identification

As with most compressive sensing and sparse approximation problems we will see that identifying the function $f$'s support $\mathcal{S}$ is the most difficult part of recovering $f$. As a result our proposed iterative algorithm spends the vast majority of its time in every iteration recovering as many energetic $\boldsymbol{n} = (n_0, n_1, \cdots, n_{D-1}) \in \mathcal{S}$ as it can. Only after doing so does it then approximate a sparse vector $\boldsymbol{c} \in \mathbb{C}^{[M]^D}$ containing nonzero coefficients $c_{\boldsymbol{n}}$ for each discovered $\boldsymbol{n} \in \mathcal{S}$. Here, each $\boldsymbol{n}$ will be referred to as an *index vector* of an entry in $\boldsymbol{c}$. Let $\operatorname{supp}(\boldsymbol{v}) \subseteq [M]^D$ represent the set of index vectors whose corresponding $v_{\boldsymbol{n}}$ entries are nonzero. We introduce the following notation in order to help explain our algorithm in the subsequent sections of the chapter.

For a given $\boldsymbol{v} \in \mathbb{C}^{[M]^D}$, $j \in [D]$, and $\widetilde{n} \in [M]$ the vector $\boldsymbol{v}_{j;\widetilde{n}} \in \mathbb{C}^{[M]^{D-1}}$ indexed by $\boldsymbol{k} \in [M]^{D-1}$ is defined by

$$\left(v_{j;\widetilde{n}}\right)_{\boldsymbol{k}} = \begin{cases} v_{\boldsymbol{n}}, & \text{if } \boldsymbol{n} = (k_0, \ldots, k_{j-1}, \widetilde{n}, k_j, \ldots, k_{D-2}) \\ 0 & \text{otherwise} \end{cases}. \tag{4.7}$$

Note that $\boldsymbol{v}_{j;\widetilde{n}}$ will only ever have at most

$$N' := \binom{D-1}{d-\|\widetilde{n}\|_0} M^{\min\{d-\|\widetilde{n}\|_0, D-1\}} \leq \left(\frac{e(D-1)M}{\max\{d-1,1\}}\right)^d \tag{4.8}$$

nonzero entries if $v_{\boldsymbol{n}} = 0$ for all $\boldsymbol{n} \in [M]^D$ with $\|\boldsymbol{n}\|_0 > d$ by assumption. Here, as throughout the remainder of the chapter, we define $\binom{p}{q}$ to be 1 whenever $q \geq p$ or $q < 0$ (also recall the definition of $\|\widetilde{n}\|_0$ from (4.5) above).[4] Similarly, for a given $\boldsymbol{v} \in \mathbb{C}^{[M]^D}$, $j \in [D]$, and $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$ the vector $\boldsymbol{v}_{j;(\widetilde{\boldsymbol{n}},\cdots)} \in \mathbb{C}^{[M]^{D-j-1}}$ indexed by $\boldsymbol{k} \in [M]^{D-j-1}$ is defined by

$$\left(v_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right)_{\boldsymbol{k}} = \begin{cases} v_{\boldsymbol{n}}, & \text{if } \boldsymbol{n} = (\widetilde{\boldsymbol{n}}, \boldsymbol{k}) \\ 0 & \text{otherwise} \end{cases}. \tag{4.9}$$

The following lemma bounds the total number of nonzero entries that $\boldsymbol{v}_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ can have given that $v_{\boldsymbol{n}} = 0$ whenever $\|\boldsymbol{n}\|_0 > d$. Note that for $j = 0$, $\boldsymbol{v}_{j;\widetilde{n}} = \boldsymbol{v}_{j;(\widetilde{n},\cdots)}$ so that (4.8) follows as a special case.

**Lemma 1.** *Let $\boldsymbol{v} \in \mathbb{C}^{[M]^D}$, $j \in [D]$, and $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$ with $\|\widetilde{\boldsymbol{n}}\|_0 \leq d$. Suppose that $v_{\boldsymbol{n}} = 0$ whenever $\|\boldsymbol{n}\|_0 > d$. Then $v_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ can have at most*

$$\widetilde{N}_j := \binom{D-j-1}{d-\|\widetilde{\boldsymbol{n}}\|_0} M^{\min\{d-\|\widetilde{\boldsymbol{n}}\|_0, D-j-1\}} \leq \left(\frac{e(D-j-1)M}{\max\{d-j-1,1\}}\right)^d \tag{4.10}$$

*nonzero entries.*

*Proof.* Since $v_{\boldsymbol{n}} = 0$ whenever $\|\boldsymbol{n}\|_0 > d$ it must be the case that $\left(v_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right)_{\boldsymbol{n}} = 0$ whenever

[4] The $\min\{d - \|\widetilde{n}\|_0, D-1\}$ in the exponent of the $M$ in (4.8) handles the case when $d = D$ and $\widetilde{n} = 0$.

$\boldsymbol{n} = (\widetilde{\boldsymbol{n}}, \widetilde{\boldsymbol{n}}')$ has $\|\widetilde{\boldsymbol{n}}'\|_0 > d - \|\widetilde{\boldsymbol{n}}\|_0$, where $\widetilde{\boldsymbol{n}}' \in \mathbb{C}^{D-j-1}$. As a result, if $D-j-1 > d - \|\widetilde{\boldsymbol{n}}\|_0$ then there are at most $\binom{D-j-1}{d-\|\widetilde{\boldsymbol{n}}\|_0}$ entry combinations left in $\boldsymbol{n}$ which can be nonzero, each of which can take on $M$ different values. If, on the other hand, $d - \|\widetilde{\boldsymbol{n}}\|_0 \geq D-j-1$ then all of the remaining $D-j-1$ values of $\boldsymbol{n}$ can each take on $M$ different values. $\square$

Motivated by the definition of $\boldsymbol{v}_{j;\widetilde{n}}$ in (4.7) we further define

$$\mathcal{I}_{j;\widetilde{n}} := \left\{\boldsymbol{n} \in \mathcal{I} \mid n_j = \widetilde{n}\right\} \subset \mathbb{C}^{[M]^D},$$

and denote the restriction matrix that projects vectors in $\mathbb{C}^{[M]^D}$ onto each $\mathcal{I}_{j;\widetilde{n}}$ (considered as a subset of $\mathbb{C}^{[M]^{D-1}}$) by $P_{j;\widetilde{n}} \in \{0,1\}^{M^{D-1} \times M^D}$. That is, we consider each $P_{j;\widetilde{n}}$ matrix to have rows indexed by $\boldsymbol{l} \in [M]^{D-1}$, columns indexed by $\boldsymbol{k} \in [M]^D$, and entries defined by

$$(P_{j;\widetilde{n}})_{\boldsymbol{l},\boldsymbol{k}} := \begin{cases} 1 & \text{if } \boldsymbol{k} = (l_0, \ldots, l_{j-1}, \widetilde{n}, l_j, \ldots, l_{D-2}) \\ 0 & \text{otherwise} \end{cases}. \tag{4.11}$$

As a result, we have that $P_{j;\widetilde{n}}\boldsymbol{v} = \boldsymbol{v}_{j;\widetilde{n}}$ for all $\boldsymbol{v} \in \mathbb{C}^{[M]^D}$.

The fast support identification strategy we will employ in this chapter will effectively boil down to rapidly approximating the norms of various $\boldsymbol{c}_{j;\widetilde{n}}$ and $\boldsymbol{c}_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ vectors for carefully chosen collections of $\widetilde{n} \in [M]$ and $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$. This, in turn, will be done using as few evaluations of $f$ in (4.6) as possible in order to estimate inner products and norms of other proxy functions constructed from $f$. As a simple example, note that $\|\boldsymbol{c}\|_2$ can be estimated by using samples from $f$ in order to approximate $\|f\|^2_{L^2(\mathcal{D},\boldsymbol{\nu})}$ since

$$\|f\|^2_{L^2(\mathcal{D},\boldsymbol{\nu})} = \sum_{\boldsymbol{n}\in[M]^D} |c_{\boldsymbol{n}}|^2.$$

A bit less trivially, for $j \in [D]$ and $\widetilde{n} \in [M]$ one can also define the function $\left\langle f, T_{j;\widetilde{n}} \right\rangle_{(\mathcal{D}_j,\nu_j)}$ : $\mathcal{D}'_j \to \mathbb{C}$ with domain $\mathcal{D}'_j := \times_{k \neq j} \mathcal{D}_k$ by having $\left\langle f, T_{j;\widetilde{n}} \right\rangle_{(\mathcal{D}_j,\nu_j)}(\boldsymbol{w})$ evaluate to

$$\int_{\mathcal{D}_j} f(w_0, \ldots, w_{j-1}, z, w_{j+1}, \ldots, w_{D-2}) \overline{T_{j;\widetilde{n}}(z)} \; d\nu_j(z)$$

for all $\boldsymbol{w} \in \mathcal{D}'_j$. Let $\boldsymbol{\nu}'_j := \otimes_{k \neq j} \nu_k$. It is not too difficult to see that

$$\left\| \langle f, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j,\nu_j)} \right\|^2_{L^2(\mathcal{D}'_j, \boldsymbol{\nu}'_j)} = \sum_{\boldsymbol{n} \text{ s.t. } n_j = \widetilde{n}} |c_{\boldsymbol{n}}|^2 = \|\boldsymbol{c}_{j;\widetilde{n}}\|_2^2 \tag{4.12}$$

in this case. Similarly, for some $j \in [D]$ and $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$ one can define the function $\langle f, T_{j;\widetilde{\boldsymbol{n}}} \rangle_{\left(\times_{i \in [j+1]} \mathcal{D}_i, \otimes_{i \in [j+1]} \nu_i\right)}$ from $\mathcal{D}''_j := \times_{k > j} \mathcal{D}_k$ into $\mathbb{C}$ by letting $\langle f, T_{j;\widetilde{\boldsymbol{n}}} \rangle_{\left(\times_{i \in [j+1]} \mathcal{D}_i, \otimes_{i \in [j+1]} \nu_i\right)}(\boldsymbol{w})$ equal to

$$\int_{\times_{i \in [j+1]} \mathcal{D}_i} f(\boldsymbol{z}, w_0, \ldots, w_{D-j-2}) \prod_{k \in [j+1]} \overline{T_{k;\widetilde{n}_k}(z_k)} \; d\left(\otimes_{i \in [j+1]} \nu_i\right)(\boldsymbol{z})$$

for all $\boldsymbol{w} \in \mathcal{D}''_j$. Let $\boldsymbol{\nu}''_j := \otimes_{k > j} \nu_k$. Analogously to the situation above we then have that

$$\left\| \langle f, T_{j;\widetilde{\boldsymbol{n}}} \rangle_{\left(\times_{i \in [j+1]} \mathcal{D}_i, \otimes_{i \in [j+1]} \nu_i\right)} \right\|^2_{L^2\left(\mathcal{D}''_j, \boldsymbol{\nu}''_j\right)} = \|\boldsymbol{c}_{j;(\widetilde{\boldsymbol{n}}, \cdots)}\|_2^2. \tag{4.13}$$

As we shall see below, both (4.12) and (4.13) will be implicitly utilized in order to allow the estimation of such $\|\boldsymbol{c}_{j;\widetilde{n}}\|_2^2$ and $\|\boldsymbol{c}_{j;(\widetilde{\boldsymbol{n}}, \cdots)}\|_2^2$ norms, respectively, using just a few nonadaptive samples from $f$.

### 4.1.6 The Proposed Method as a Sublinear-Time Compressive Sensing Algorithm

Note that $\boldsymbol{c}$, $\boldsymbol{c}_{j;\widetilde{n}}$ and $\boldsymbol{c}_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ from (4.6) are all at most $s$-sparse under the assumption that $\epsilon = 0$. As mentioned above, this means that recovering $f$ from a few function evaluations is essentially equivalent to recovering $\boldsymbol{c}$ using random sampling matrices. Given this, the method we propose in the next section can also be viewed as a sublinear-time compressive sensing algorithm which uses a highly structured measurement matrix $A$ consisting of several concatenated random sampling matrices. More explicitly, the measurements $\boldsymbol{y} \in \mathbb{C}^{m'_1+m'_2+m'_3}$ utilized by Algorithm 5 below consist of function evaluations (i.e., recall (4.4)) which can be represented in the concatenated form

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^{\mathbf{E}} \\ \boldsymbol{y}^{\mathbf{I}} \\ \boldsymbol{y}^{\mathbf{P}} \end{bmatrix} = \begin{bmatrix} \Phi\boldsymbol{c} \\ A^I\boldsymbol{c} \\ A^P\boldsymbol{c} \end{bmatrix} = \begin{bmatrix} \Phi \\ A^I \\ A^P \end{bmatrix} \boldsymbol{c} = A\boldsymbol{c} \tag{4.14}$$

for subvectors $\boldsymbol{y}^{\mathbf{E}} \in \mathbb{C}^{m'_1}$, $\boldsymbol{y}^{\mathbf{I}} \in \mathbb{C}^{m'_2}$, $\boldsymbol{y}^{\mathbf{P}} \in \mathbb{C}^{m'_3}$ (recall Section 4.1.3), and structured sampling matrices $\Phi \in \mathbb{C}^{m'_1 \times M^D}$, $A^I \in \mathbb{C}^{m'_2 \times M^D}$, and $A^P \in \mathbb{C}^{m'_3 \times M^D}$.

In (4.14) the matrix $\Phi$ is a standard random sampling matrix with the RIP formed as per (4.3). It and its associated samples $\boldsymbol{y}^{\mathbf{E}} = \Phi\boldsymbol{c}$ are used to estimate the entries of $\boldsymbol{c}$ indexed by the index vectors contained in the identified energetic support set $T$ in line 13 of Algorithm 5. The matrices $A^I$ and $A^P$ are both used for support identification. In particular, $A^I$ and its associated samples $\boldsymbol{y}^{\mathbf{I}} = A^I\boldsymbol{c}$ are used to try to identify all of the energetic basis functions in each input dimension, i.e., the sets

$$\mathcal{N}'_j := \left\{\widetilde{n} \in [M] \;\middle|\; \exists \boldsymbol{n} \in \mathcal{S} \text{ with } n_j = \widetilde{n}\right\} \subseteq [M]$$

for each $j \in [D]$ (see Algorithm 6). The matrix $A^P$ and its associated samples $\boldsymbol{y}^{\mathbf{P}} = A^P\boldsymbol{c}$ are then used in Algorithm 7 to help build up the estimated support set $T \subset [M]^D$ from the previously identified $\mathcal{N}'_j$-sets. See Section 4.2 below for additional details.

The matrix $A^I$ above is built using the matrix Kronecker products $\widetilde{A}_j \otimes L_j$ for $j \in [D]$, where $\widetilde{A}_j \in \mathbb{C}^{m \times [M]^{D-1}}$ is the random sampling matrix defined in (4.20), and $L_j \in \mathbb{C}^{\mathcal{L}'_j \times [M]}$ is a sampling matrix associated with $\mathcal{U}_j \subset \mathcal{D}_j$ from Section 4.1.3 defined as

$$\left(L_j\right)_{q,n} := T_{j;n}(u_{j,q}), \qquad q \in [\mathcal{L}'_j] \text{ and } n \in [M]. \tag{4.15}$$

The matrix $A^P$, on the other hand, is constructed using $B_j \otimes C_j$ for all $j \in [D] \setminus \{0\}$ where each $B_j \in \mathbb{C}^{m_1 \times [M]^{j+1}}$ is the random sampling matrix defined below in (4.35), and each $C_j \in \mathbb{C}^{m_2 \times [M]^{D-j-1}}$ the random sampling matrix defined in (4.34). In particular, we have that

$$A^I := \begin{bmatrix} \widetilde{A}_0 \otimes L_0 \\ \vdots \\ \widetilde{A}_{D-1} \otimes L_{D-1} \end{bmatrix}, \text{ and } A^P := \begin{bmatrix} B_1 \otimes C_1 \\ \vdots \\ B_{D-1} \otimes C_{D-1} \end{bmatrix}. \tag{4.16}$$

From a set of random gaussian weights $\mathcal{W}$, a marix $G \in \mathbb{C}^{L \times m}$ is defined as

$$(G)_{k,\ell} := g_\ell^k, \qquad k \in [L] \text{ and } \ell \in [m], \tag{4.17}$$

where $g_\ell^k$'s are the gaussian weights from (4.19).

Briefly contrasting the proposed approach interpreted as a sublinear-time compressive sensing method via (4.14) against previously existing sublinear-time algorithms for Compressive Sensing (CS) (see, e.g., [38, 39, 47, 48, 49]), we note that no previous sublinear-time CS methods exist which utilize measurement matrices solely derived from general BOS random sampling matrices. This means that the associated recovery algorithms developed herein can not directly take advantage of the standard group testing, hashing, and error correcting code-based techniques which have been regularly employed by such methods, making the development of fast reconstruction techniques and their subsequent analysis quite challenging. Nonetheless, we will see that we can still utilize at least some of the core ideas of these methods by sublinearizing the runtime of one of their well known superlinear-time relatives, CoSaMP [22].

## 4.2 The Proposed Method

In this section we introduce and discuss our proposed method. Roughly speaking, our algorithm can be considered as a greedy pursuit algorithm (see, e.g., [50, 23, 22, 51, 52]) with a faster support identification technique that takes advantage of the structure of BOS product bases. In particular, we will focus on the CoSaMP algorithm [22] herein. Note that support identification is the most computationally expensive step of the CoSaMP algorithm. Otherwise, CoSaMP is already a sublinear-time method for any type of BOS basis one likes. Our overall strategy, therefore, will be to hijack the CoSaMP algorithm as well as its analysis by removing its superlinear-time support identification procedure and replacing it with a new sublinear-time version that still satisfies the same iteration invariant as the original. See Algorithm 5 for pseudocode of our modified CoSaMP method. Note that most its steps

**Algorithm 5** Sublinearized CoSaMP

```
1: procedure SublinearRecoveryAlgorithm
2: Input: Sampling matrices $\Phi$, $A^I$, $A^P$ (implicitly, via the samples in $\mathcal{G}$ that determine
   their rows), samples $\boldsymbol{y}^{\mathbf{E}} = \Phi \boldsymbol{c}$, $\boldsymbol{y}^{\mathbf{I}} = A^I \boldsymbol{c}$, $\boldsymbol{y}^{\mathbf{P}} = A^P \boldsymbol{c}$, a sparsity estimate $s$, and a set
   of i.i.d. Gaussian weights $\mathcal{W}$
3: Output: $s$-sparse approximation $\boldsymbol{a}$ of $\boldsymbol{c}$
4:     $\boldsymbol{a}^0 = \mathbf{0}$                                                  {Initial approximation}
5:     $\boldsymbol{v}^I \leftarrow \boldsymbol{y}^{\mathbf{I}}$, $\boldsymbol{v}^P \leftarrow \boldsymbol{y}^{\mathbf{P}}$
6:     $t \leftarrow 0$
7:     repeat
8:         {The next line calls Algorithm 6 ...}
9:         $\mathcal{N}_j\ \forall j \in [D] \leftarrow$ EntryIdentification($\boldsymbol{v}^I$, $\mathcal{W}$)  {Support identification step # 1}
10:        {The next line calls Algorithm 7 ...}
11:        $\Omega \leftarrow$ Pairing($\boldsymbol{v}^P$, $\mathcal{N}_j\ \forall j \in [D]$)          {Support identification step # 2}
12:        $T \leftarrow \Omega \cup \operatorname{supp}(\boldsymbol{a}^t)$                                 {Merge supports}
13:        $\boldsymbol{b}_T \leftarrow \Phi_T^{\dagger} \boldsymbol{y}^{\mathbf{E}}$                          {Local estimation by least-squares}
14:        $t \leftarrow t + 1$
15:        $\boldsymbol{a}^t \leftarrow \boldsymbol{b}_s$                               {Prune to obtain next approximation}
16:        $\boldsymbol{v}^I \leftarrow \boldsymbol{y}^{\mathbf{I}} - A^I \boldsymbol{a}^t$, $\boldsymbol{v}^P \leftarrow \boldsymbol{y}^{\mathbf{P}} - A^P \boldsymbol{a}^t$           {Update current samples}
17:    until halting criterion true
18: end procedure
```

are identical to the original CoSaMP algorithm except for the two "Support identification" steps, and the "Update current samples" and "halting criterion" lines. Thus, our discussion will mainly focus on these three parts.

Like CoSaMP, Algorithm 5 is a greedy approximation technique which makes locally optimal choices during each iteration. In the $t$-th iteration, it starts with an $s$-sparse approximation $\boldsymbol{a}^t$ of $\boldsymbol{c}$ and then tries to approximate the at most $2s$-sparse residual vector $\boldsymbol{r} := \boldsymbol{c} - \boldsymbol{a}^t$. The two "Support identification" steps begin approximating $\boldsymbol{r}$ by finding a support set $\Omega \subset \mathcal{I}$ of cardinality at most $2s$ which contains the set $\left\{ \boldsymbol{n} \;\middle|\; |r_{\boldsymbol{n}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$ (i.e., $\Omega$ contains the indices of the entries where most of the energy of $\boldsymbol{r}$ is located). These support identification steps constitute the main modification made to CoSaMP in this chapter and are discussed in more detail in sections 4.2.1 and 4.2.2 below. After support identification, in the "Merge supports" step, a new support set $T$ of cardinality at most $3s$ is then formed from the union of $\Omega$ with the support of the current approximation $\boldsymbol{a}^t$. At this stage $T$ should contain the overwhelming majority of the important (i.e., energetic) index vectors in $\mathcal{S}$. As a result, restricting the columns of the sampling matrix $\Phi$ to those in $T$ (or constructing them on the fly in a low memory setting) in order to solve for $\boldsymbol{b}_T := \operatorname{argmin}_{\boldsymbol{u} \in \mathbb{C}^{|T|}} \|\Phi_T \boldsymbol{u} - \boldsymbol{y}^{\mathbf{E}}\|_2$ should yield accurate estimates for the true coefficients of $\boldsymbol{c}$ indexed by the elements of $T$, $\boldsymbol{c}_T$.[5] The vector $\boldsymbol{b}_s$ restricting $\boldsymbol{b}_T$ to its $s$ largest-magnitude elements then becomes the next approximation of $\boldsymbol{c}$, $\boldsymbol{a}^{t+1}$.

As previously mentioned, the main difference between Algorithm 5 and CoSaMP is in the support identification steps. In the proposed method support identification consists of two parts: "Entry Identification" and "Pairing". For each of these steps we use a different measurement matrix, $A^I$ or $A^P$, respectively, as well as a different set of samples (either $\boldsymbol{v}^I$ or $\boldsymbol{v}^P$) from the current residual vector. Thus, we need to update a total of three estimates every iteration: $\boldsymbol{v}^I$, $\boldsymbol{v}^P$ and $\boldsymbol{a}^t$. In the next two sections we review each of the two newly proposed support identification steps in more detail.

[5]In practice, it suffices to approximate the least-squares solution $b_T$ by an iterative least-squares approach such as Richardson's iteration or conjugate gradient [14] since computing the exact least squares solution can be expensive when $s$ is large. The argument of [22] shows that it is enough to take three iterations for Richardson's iteration or conjugate gradient if the initial condition is set to $\boldsymbol{a}^t$, and if $\Phi$ has an RIP constant $\delta_{2s} < 0.025$. In fact, both of these methods have similar runtime performance.

### 4.2.1 Support Identification Step # 1: Entry Identification

**Algorithm 6** Entry identification

```
1: procedure EntryIdentification
2: Input: $\boldsymbol{v}^I$, and a set of i.i.d. Gaussian weights $\mathcal{W}$ for use in the $h_{j;k}$ below (see (4.19))
3: Output: $\mathcal{N}_j$ for $j \in [D]$
4:     for $j = 0 \to D-1$ do
5:         $\mathcal{N}_j \leftarrow \emptyset$
6:         for $\widetilde{n} = 0 \to M-1$ do
7:             {The method works because $\text{median}_{k\in[L]} \left| \langle h_{j;k}, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j,\nu_j)} \right| \approx \|\boldsymbol{r}_{j;\widetilde{n}}\|_2$. See
8:             (4.18) and (4.19) for the definition of $h_{j;k}$. For exactly $s$-sparse $\boldsymbol{c}$ one can use
9:             $\tau = 0$ below. More generally, one can select the largest $s$ estimates for $\mathcal{N}_j$.}
10:            if $\text{median}_{k\in[L]} \left| \langle h_{j;k}, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j,\nu_j)} \right|$, then
11:                $\mathcal{N}_j \leftarrow \{\widetilde{n}\} \cup \mathcal{N}_j$.
12:            end if
13:        end for
14:    end for
15: end procedure
```

For each $j \in [D]$ the entry identification algorithm (see Algorithm 6) tries to find the $j$-th entry of each energetic index vector $\boldsymbol{n}$ corresponding to a nonzero entry $r_{\boldsymbol{n}}$ in the $2s$-sparse residual vector $\boldsymbol{r} = \boldsymbol{c} - \boldsymbol{a}^t$. Note that for each $j$ this gives rise to at most $2s$ index entries in $[M]$.[6] We therefore define $\mathcal{N}_j^t$ to be the resulting set $\{n_j \mid \boldsymbol{n} \in \text{supp}(\boldsymbol{r})\}$ of size at most

[6]Note that we are generally assuming herein that $2s < M$. In the event that $2s \geq M$ one can proceed in at least two different ways. The first way is to not change anything, and to simply be at peace with the possibility of, e.g., occasionally returning $\mathcal{N}_j = [M]$. This is our default approach. The second way is regroup the first $g \in \mathbb{N}$ variables of $f$ together into a new collective "first variable", the second $g$ variables together into a new collective "second variable", etc., for some $g$ satisfying $M^g > 2s$. After such regrouping

$2s$ for each $j \in [D]$. Note that $\widetilde{n} \in \mathcal{N}_j^t$ if and only if $\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 > 0$. As a result we can learn $\mathcal{N}_j^t$ by approximating $\|\boldsymbol{r}_{j;\widetilde{n}}\|_2$ using $\left\| \langle h, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j, \nu_j)} \right\|^2_{L^2(\mathcal{D}'_j, \boldsymbol{\nu}'_j)}$ via (4.12) as long as we know

$$h(\boldsymbol{x}) := \sum_{\boldsymbol{n} \in \mathrm{supp}(\boldsymbol{r})} r_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x}). \tag{4.18}$$

Whenever $\left\| \langle h, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j, \nu_j)} \right\|^2_{L^2(\mathcal{D}'_j, \boldsymbol{\nu}'_j)}$ is larger than a threshold value (e.g., zero) for a particular choice of $j \in [D]$ and $\widetilde{n} \in [M]$, we could simply add $\widetilde{n}$ to $\mathcal{N}_j$ (our estimate of $\mathcal{N}_j^t$) in this case.

Of course we don't actually know exactly what $h$ is. However, we do have access to samples from $h$ in each iteration in the form of $\boldsymbol{v}^I$ and $\boldsymbol{v}^P$. And, as a result, we are able to approximate $\left\| \langle h, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j, \nu_j)} \right\|^2_{L^2(\mathcal{D}'_j, \boldsymbol{\nu}'_j)} = \|\boldsymbol{r}_{j;\widetilde{n}}\|_2$ for any $j \in [D]$ and $\widetilde{n} \in [M]$ with the estimator

$$\mathrm{median}_k \left| \langle h_{j;k}, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j, \nu_j)} \right|$$

defined using (4.19). In section 4.3.1 we show that this estimator can be used to accurately approximate $\|\boldsymbol{r}_{j;\widetilde{n}}\|_2$ *for all* $2s$*-sparse residual vectors* $\boldsymbol{r} \in \mathbb{C}^{[M]^D}$ using only $\mathcal{O}\big(s \cdot K^2 d \mathcal{L}' \cdot \mathrm{polylog}(D, s, M, K)\big)$ universal samples from any given $\boldsymbol{r}$'s associated $h$-function in (4.18) (i.e., the samples in $\boldsymbol{v}^I$).[7] Furthermore, the estimator can always be computed in just $\mathcal{O}\left(s^2 \cdot K^2 d^2 \mathcal{L} \cdot \mathrm{polylog}(D, s, M, K)\right)$-time.

At this point it is important to note that managing to find each $\mathcal{N}_j^t$ exactly for all $j \in [D]$ still does not provide enough information to allow us to learn $\mathrm{supp}(\boldsymbol{r})$ when $d > 1$. In general the most we learn from this information is that $\mathrm{supp}(\boldsymbol{r}) \subset \left( \times_{j \in [D]} \mathcal{N}_j \right) \bigcap \mathcal{I} \subset [M]^D$. In

the algorithm can then again effectively be run as is with respect to these new collective variables.

[7]Recall that $\mathcal{L}'$ represents the maximum number of function evaluations one needs in order to compute $\langle g, T_{j;\widetilde{n}} \rangle$ for all $\widetilde{n} \in [M]$ in $\mathcal{O}(\mathcal{L})$-time for any given $j \in [D]$, and $s$-sparse $g : \mathcal{D}_j \to \mathbb{C}$ in $\mathrm{span}\left\{ T_{j;m} \mid m \in [M] \right\}$.

the next "Pairing" step we address this problem by iteratively pruning the candidates in $\times_{j\in[1]}\mathcal{N}_j, \times_{j\in[2]}\mathcal{N}_j, \dots, \times_{j\in[D]}\mathcal{N}_j$ down at each stage to the best $2s$ candidates for being a prefix of some element in $\operatorname{supp}(\boldsymbol{r})$. As we shall see, the pruning in each "Pairing" stage involves energy estimates that are computed using only the samples from $h$ in $\boldsymbol{v}^P$. These ideas are discussed in greater detail in the next section.

### 4.2.2 Support Identification Step # 2: Pairing

**Algorithm 7** Pairing

```
1: procedure Pairing
2: Input: $\boldsymbol{v}^P = \{h(\boldsymbol{w}_{j,\ell}, \boldsymbol{z}_{j,k}) \mid j \in [D] \setminus \{0\}, \ell \in [m_1], k \in [m_2]\}$, $\mathcal{N}_j$ for $j \in [D]$
3: Output: $\mathcal{P}$
4:     $\mathcal{P}_0 \leftarrow \mathcal{N}_0$
5:     for $j = 1 \to D-1$ do
6:         {This method works because $E_{j;(\widetilde{\boldsymbol{n}}\dots)} \approx \|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2$ below.}
7:         $E_{j;(\widetilde{\boldsymbol{n}}\dots)} \leftarrow \frac{1}{m_2}\sum_{k\in[m_2]}\left|\frac{1}{m_1}\sum_{\ell\in[m_1]} h(\boldsymbol{w}_{j,\ell}, \boldsymbol{z}_{j,k})\overline{T_{\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_{j,\ell})}\right|^2 \ \forall \widetilde{\boldsymbol{n}} \in \mathcal{P}_{j-1} \times \mathcal{N}_j$.
8:         Create $\mathcal{P}_j$ containing each $\widetilde{\boldsymbol{n}}$ whose energy estimate $E_{j;(\widetilde{\boldsymbol{n}}\dots)}$ is in the $2s$-largest.
9:     end for
10:    $\mathcal{P} \leftarrow \mathcal{P}_{D-1}$
11: end procedure
```

Once all the $\mathcal{N}_j \subset [M]$ have been identified for all $j \in [D]$ it remains to match them together in order learn the true length-$D$ index vectors in $\operatorname{supp}(\boldsymbol{r}) \subset [M]^D$. To achieve this we begin by attempting to identify all the prefixes of length two, $\widetilde{\boldsymbol{n}} = (\widetilde{n}_0, \widetilde{n}_1) \in \mathcal{N}_0 \times \mathcal{N}_1$, which begin at least one element in the support of $\boldsymbol{r}$. Similar to the ideas utilized above, we now note that $(\widetilde{\boldsymbol{n}}, \boldsymbol{n}') \in \text{supp}(\boldsymbol{r})$ for some $\boldsymbol{n}' \in [M]^{D-2}$ if and only if $\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2 > 0$. As a result, it suffices for us to use the samples from $h$ (recall (4.18)) in $\boldsymbol{v}^P$ in order to compute $E_{j;(\widetilde{\boldsymbol{n}}\dots)} \approx \|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2$ in Algorithm 7 above. The $2s$-largest estimates $E_{j;(\widetilde{\boldsymbol{n}}\dots)}$ are then used to identify all the prefixes of length 2 which begin at least one element of $\text{supp}(\boldsymbol{r})$. Of course, this same idea can then be used again to find all length-3 prefixes of elements in $\text{supp}(\boldsymbol{r})$ by extending the previously identified length-2 prefixes in all possible $\mathcal{O}(s^2)$ combinations with the elements in $\mathcal{N}_2$, and then testing the resulting length-3 prefixes' energies in order to identify the $2s$ most energetic such combinations, etc.. See Algorithm 7 above for pseudocode, Section 4.3.2 for analysis of these $E_{j;(\widetilde{\boldsymbol{n}}\dots)}$ estimators, and Section 4.2.2.1 just below for a concrete example of the pairing process.

#### 4.2.2.1 An Example of Entry Identification and Pairing to Find Support

Assume that $\boldsymbol{r} \in \mathbb{C}^{[M]^3}$ is three-sparse with a priori unknown energetic index vectors

$$\text{supp}(\boldsymbol{r}) = \{(3,5,6),\ (4,7,8),\ (11,5,100)\} \subset [M]^3$$

and corresponding nonzero coefficients $r_{(3,5,6)}$, $r_{(4,7,8)}$, and $r_{(11,5,100)}$. We can further imagine that $M$ here is significantly larger than, e.g., 100 so that computing all $M^3$ coefficients of $\boldsymbol{r}$ using standard numerical methods would be undesirable. In this case, Algorithm 6 aims to output the sets

$$\mathcal{N}_0 = \{3,4,11\},\ \mathcal{N}_1 = \{5,7\},\ \text{and}\ \mathcal{N}_2 = \{100,6,8\} \subset [M],$$

i.e., the first, second, and third entries of each index vector in the support of $\boldsymbol{r}$, respectively. Note that there are $18 = 3 \times 2 \times 3$ possible index vectors which are consistent with the

$\mathcal{N}_0$, $\mathcal{N}_1$, and $\mathcal{N}_2$ above. Algorithm 7 is now tasked with finding out which of these 18 possibilities are truly elements of supp($\boldsymbol{r}$) without having to test them all individually.[8]

To identify supp($\boldsymbol{r}$) without having to test all 18 index vectors in $\mathcal{N}_0 \times \mathcal{N}_1 \times \mathcal{N}_2$, the pairing process instead starts by estimating the energy of the $|\mathcal{N}_0| \cdot |\mathcal{N}_1| = 6$ length-2 prefixes in $\mathcal{N}_0 \times \mathcal{N}_1$ which might begin an index vector in supp($\boldsymbol{r}$). In the ideal case these energy estimates will reveal that only 3 of these 6 possible length-2 prefixes actually have any energy,

$$\mathcal{P}_1 = \{(3,5), (4,7), (11,5)\} \subset [M]^2.$$

In its next stage the pairing process now continues by combining these three length-2 prefixes in $\mathcal{P}_1$ with $\mathcal{N}_2$ in order to produce $|\mathcal{P}_1| \times |\mathcal{N}_2| = 9$ final candidate elements potentially belonging to supp($\boldsymbol{r}$) $\subset [M]^3$. Estimating the energy of these 9 candidates then finally reveals the true identities of the index vectors in the support of $\boldsymbol{r}$.

Note that instead of computing energy estimates for all 18 possible support candidates in $\mathcal{N}_0 \times \mathcal{N}_1 \times \mathcal{N}_2$, the pairing process allows us to determine the correct support of $\boldsymbol{r}$ using only $15 = 6 + 9 < 18$ total energy estimates in this example. Though somewhat underwhelming in this particular example, the improvement provided by Algorithm 7 becomes much more significant as the dimension $D$ of the index vectors grows larger. When $|\text{supp}(\boldsymbol{r})| = |\mathcal{N}_j| = s$ for all $j \in [D]$, for example, $\times_{j \in [D]} \mathcal{N}_j$ will have $s^D$ total elements. Nonetheless, Algorithm 7 will be able to identify supp($\boldsymbol{r}$) using only $\mathcal{O}\left(s^2 D\right)$ energy estimates in the ideal setting.[9]

[8]In this simple example we can of course simply estimate the energy for all 18 possible index vectors. The three true index vectors in the support of $\boldsymbol{r}$ with nonzero energy would then be discovered and all would be well. However, this naive approach becomes spectacularly inefficient for larger $D \gg 3$.

[9]In less optimal settings one should keep in mind that Algorithm 7 only finds the most energetic entries in general, so that $\mathcal{P} \supset \left\{ \boldsymbol{n} \mid |r_{\boldsymbol{n}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$ for a given $\alpha > 1$. This is why we need to apply it iteratively.

### 4.2.3 A Theoretical Guarantee for Support Identification

The following lemma and theorem show that our support identification procedure (i.e., Algorithm 6, followed by Algorithm 7) always identifies the indexes of the majority of the energetic entries in $\boldsymbol{r}$. Consequently, the energy of the residual is guaranteed to decrease from iteration to iteration of Algorithm 5. We want to remind readers that $\boldsymbol{r}$ is always $2s$-sparse since $\boldsymbol{c}$ and each $\boldsymbol{a}^{t-1}$ are $s$-sparse in the present analysis (i.e., $\epsilon = 0$).

**Lemma 2.** *Suppose that $\left\{T_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathcal{I} \subseteq [M]^D\right\}$ is a BOS where each basis function $T_{\boldsymbol{n}}$ is defined as per* (4.1)*. Let $\mathcal{H}_{2s}$ be the set of all functions, $h : \mathcal{D} \to \mathbb{C}$, in $\operatorname{span}\left\{T_{\boldsymbol{n}} \mid \boldsymbol{n} \in \mathcal{I}\right\}$ whose coefficient vectors are $2s$-sparse, and let $\boldsymbol{r}_h \in \mathbb{C}^{\mathcal{I}}$ denote the $2s$-sparse coefficient vector for each $h \in \mathcal{H}_{2s}$. Fix $p \in (0, 1/2)$, $1 \le d \le D$, $N = \binom{D}{d} M^d$, and $K = \sup_{\boldsymbol{n} \text{ s.t.} \|\boldsymbol{n}\|_0 \le d} \|T_{\boldsymbol{n}}\|_\infty$. Then, one can randomly select a set of i.i.d. Gaussian weights $\mathcal{W} \subset \mathbb{R}$ for use in* (4.19)*, and also randomly construct entry identification and pairing grids, $\mathcal{G}^I \subset \mathcal{D}$ and $\mathcal{G}^P \subset \mathcal{D}$ (recall Section 4.1.3), whose total cardinality $\left|\mathcal{G}^I\right| + \left|\mathcal{G}^P\right|$ is*
$\mathcal{O}\left(sD\mathcal{L}'K^2 \max\{\log^2(s)\log^2(DN), \log(\frac{D}{p})\} + s^3DK^4 \max\left\{\log^4(s)\log^2(N)\log^2(DN), \log^2(\frac{D}{p})\right\}\right)$, *such that the following property holds $\forall h \in \mathcal{H}_{2s}$ with probability greater than $1 - 2p$: Let $\boldsymbol{v}_h^I \in \mathbb{C}^{m_2'}$ and $\boldsymbol{v}_h^P \in \mathbb{C}^{m_3'}$ be samples from $h \in \mathcal{H}_{2s}$ on $\mathcal{G}^I$ and $\mathcal{G}^P$, respectively. If Algorithms 6 and 7 are granted access to $\boldsymbol{v}_h^I$, $\boldsymbol{v}_h^P$, $\mathcal{G}^I$, $\mathcal{G}^P$, and $\mathcal{W}$ then they will find a set $\Omega \subset [M]^D$ of cardinality $2s$ in line 11 of Algorithm 5 such that*

$$\|(\boldsymbol{r}_h)_{\Omega^c}\|_2 \le 0.202\|\boldsymbol{r}_h\|_2.$$

*Furthermore, the total runtime complexity of Algorithms 6 and 7 is always*
$\mathcal{O}\left(s^2D\mathcal{L}K^2 \max\{\log^2(s)\log^2(DN), \log(\frac{D}{p})\} + s^5D^2K^4 \max\left\{\log^4(s)\log^2(N)\log^2(DN), \log^2(\frac{D}{p})\right\}\right)$.

In Lemma 2 above $\mathcal{L}'$ denotes the maximum number of points in $\mathcal{D}_j$ required in order to determine the value of $\langle g, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}$ for all $\widetilde{n} \in [M]$ in $\mathcal{O}(\mathcal{L})$-time for any given $s$-sparse $g : \mathcal{D}_j \to \mathbb{C}$ in $\operatorname{span}\left\{T_{j;m} \;\middle|\; m \in [M]\right\}$, maximized over all $j \in [D]$. See Section 4.1.2 for a more in depth discussion of these quantities. The reader is also referred back to Section 4.1.3 for a discussion of the entry identification and pairing grids, $\mathcal{G}^I$ and $\mathcal{G}^P$, mentioned in Lemma 2. The proof of Lemma 2, which is quite long and technical, is given in Section 4.3.3. Once Lemma 2 has been established, however, it is fairly straightforward to prove that Algorithm 5 will always rapidly recover any function of $D$-variables which exhibits sparsity in a tensor product basis by building on the results in [22]. We have the following theorem.

**Theorem 7.** *Suppose that $\left\{T_{\boldsymbol{n}} \;\middle|\; \boldsymbol{n} \in \mathcal{I} \subseteq [M]^D\right\}$ is a BOS where each basis function $T_{\boldsymbol{n}}$ is defined as per* (4.1)*. Let $\mathcal{F}_s$ be the subset of all functions $f \in \operatorname{span}\left\{T_{\boldsymbol{n}} \;\middle|\; \boldsymbol{n} \in \mathcal{I}\right\}$ whose coefficient vectors are $s$-sparse, and let $\boldsymbol{c}_f \in \mathbb{C}^{\mathcal{I}}$ denote the $s$-sparse coefficient vector for each $f \in \mathcal{F}_s$. Fix $p \in (0, 1/3)$, $1 \le d \le D$, $N = \binom{D}{d} M^d$, $K = \sup\limits_{\boldsymbol{n} \text{ s.t.} \|\boldsymbol{n}\|_0 \le d} \|T_{\boldsymbol{n}}\|_\infty$, and a precision parameter $\eta > 0$. Then, one can randomly select a set of i.i.d. Gaussian weights $\mathcal{W} \subset \mathbb{R}$ for use in* (4.19)*, and also randomly construct a compressive sensing grid, $\mathcal{G} \subset \mathcal{D}$, whose total cardinality $|\mathcal{G}|$ is*

$\mathcal{O}\left(sD\mathcal{L}'K^2 \max\{\log^2(s)\log^2(DN), \log(\tfrac{D}{p})\} + s^3DK^4 \max\left\{\log^4(s)\log^2(N)\log^2(DN), \log^2(\tfrac{D}{p})\right\}\right),$

*such that the following property holds $\forall f \in \mathcal{F}_s$ with probability greater than $1 - 3p$: Let $\boldsymbol{y}$ consist of samples from $f \in \mathcal{F}_s$ on $\mathcal{G}$. If Algorithm 5 is granted access to $\boldsymbol{y}$, $\mathcal{G}$, and $\mathcal{W}$, then it will produce an $s$-sparse approximation $\boldsymbol{a}$ such that*

$$\|\boldsymbol{c}_f - \boldsymbol{a}\|_2 \le C\eta.$$

*Here $C > 0$ is an absolute constant. Furthermore, the total runtime complexity of Algorithm 5 is always*

$\mathcal{O}\Big(\Big(s^2D^2\mathcal{L}K^2\max\{\log^2(s)\log^2(DN),\log(\frac{D}{p})\}+s^5D^2K^4\max\big\{\log^4(s)\log^2(N)\log^2(DN),\log^2(\frac{D}{p})\big\}\Big)$
$\times\log\frac{\|\boldsymbol{c}_f\|_2}{\eta}\Big)$ *when $|\mathcal{G}|$ is bounded as above.*

As above, we refer the reader back to Section 4.1.2 for a discussion of the quantities $\mathcal{L}'$ and $\mathcal{L}$ appearing in Theorem 7, as well as to Section 4.1.3 for more information on the compressive sensing grid $\mathcal{G} \subset \mathcal{D}$ mentioned therein.

*Proof.* In order to analyze the support identification step, we replace Lemma 4.2 in [22] by Lemma 2, and then we obtain

$$\|\boldsymbol{c}_f - \boldsymbol{a}^{t+1}\|_2 \leq 0.5\|\boldsymbol{c}_f - \boldsymbol{a}^t\|_2$$

for each iteration $t \geq 0$, which is the same as in Theorem 2.1 in [22] provided that $f$ is exactly $s$-sparse and samples are not noisy. Except for the support identification step(s), Algorithm 5 agrees with CoSaMP, so that Lemmas 4.3 – 4.5 in [22] still directly apply to Algorithm 5. After $\mathcal{O}\left(\log\frac{\|\boldsymbol{c}_f\|_2}{\eta}\right)$ iterations, we see that the $s$-sparse approximation $\boldsymbol{a}$ therefore satisfies

$$\|\boldsymbol{c}_f - \boldsymbol{a}\|_2 \leq C\eta.$$

Since the runtime complexity of the support identification steps and the sample update process in each iteration, the total running time arises from multiplying it with the number of iterations. The number of sample points, $m_1'$, $m$, $m_1$ and $m_2$ used to define the matrices $\Phi$, $\widetilde{A}_j$, $B_j$ and $C_j$ discussed in Section 4.1.6 are all chosen to ensure that these resulting measurement matrices have the RIP. Thus, the samples of $f$ in $\boldsymbol{y}$ can be reused over as many iterations as needed. Updating the samples for the entry identification or pairing causes an extra $\mathcal{O}(sDm_2')$ and $\mathcal{O}(sDm_3')$ computations respectively which causes a $D^2$ factor instead of $D$ in the first term of runtime complexity. The probability of successful recovery for all $f \in \mathcal{F}_s$ is obtained by taking the union bound over the failure probability $p$ of $\Phi$ having $\delta_{2s} < 0.025$ via Theorem 6 together with the failure probability $2p$ of Lemma 2. □

We are now prepared to begin the process of proving Lemma 2.

## 4.3 Analysis of the Support Identification

In this section, we analyze the performance of the sublinear-time support identification technique proposed herein. First, we show in Section 4.3.1 the success of the entry identification step. Indeed, Theorems 8 and 9 show under the RIP assumption that certain one-dimensional proxy functions allow us to identify the entry with large corresponding coefficients. Lemma 6 then estimates the necessary sample complexity. In Section 4.3.2, we analyze the pairing step showing that it works uniformly for any $2s$-sparse functions in Theorem 10. Finally, in Section 4.3.3, we complete the proof of the Lemma 2 providing the complete result for the proposed support identification method.

### 4.3.1 Entry Identification

In this section, we seek to find $\mathcal{N}_j$ containing the $j$-th entries of the index vectors of the nonzero transform coefficients for all $j \in [D]$. Define $[D]' := [D] \setminus \{j\}$.

Assume without loss of generality that the number $L \in \mathbb{N}$ of proxy functions is odd. Choose $\mathcal{X}_j := \{\boldsymbol{x}_{j,\ell}\}_{\ell \in [m]}$ where each $\boldsymbol{x}_{j,\ell}$ is chosen independently at random from $\mathcal{D}_j'$ according to $d\boldsymbol{\nu}_j'$. Also, choose $\{g_1^k, \cdots, g_m^k\}_{k \in [L]}$ where each $g_\ell^k$ is an i.i.d. standard Gaussian variable $\sim \mathcal{N}(0,1)$, which forms $\mathcal{W}$ introduced in Section 4.1.3. We define a function $h_{j;k} : \mathcal{D}_j \to \mathbb{C}$ of one variable in $\mathcal{D}_j$ as follows,

$$h_{j;k}(x) := \frac{1}{\sqrt{m}} \sum_{\ell \in [m]} g_\ell^k h([x, \boldsymbol{x}_{j,\ell}]) = \frac{1}{\sqrt{m}} \sum_{\ell \in [m]} g_\ell^k \sum_{\boldsymbol{n} \in \operatorname{supp}(\boldsymbol{r})} r_{\boldsymbol{n}} T_{j;n_j}(x) \prod_{i \in [D]'} T_{i;n_i}(x_{j,\ell})_i, \tag{4.19}$$

and $[x, \boldsymbol{x}_{j,\ell}]$ is the vector obtained by inserting the variable $x$ between the entries of $\boldsymbol{x}_{j,\ell}$ indexed by $[j]$ and $\{j, j+1 \cdots, D-2\}$, i.e.,

$$[x, \boldsymbol{x}_{j,\ell}] := \big((x_{j,\ell})_0, (x_{j,\ell})_1, \cdots, (x_{j,\ell})_{j-1},\ x\ , (x_{j,\ell})_j, \cdots, (x_{j,\ell})_{D-2}\big)\,.$$

For the sake of simplicity, we let $T_{\check{\boldsymbol{n}}}(\check{\boldsymbol{x}}) := \prod_{i \in [D]'} T_{i;\check{n}_i}(\check{x}_i)$ with $\check{\boldsymbol{n}} \in [M]^{D-1}$ and $\check{\boldsymbol{x}} \in \mathcal{D}'_j$.

We choose $m$ large enough above to form the normalized random sampling matrix $\widetilde{A}_j \in \mathbb{C}^{m \times [M]^{D-1}}$ for each $j \in [D]$, defined as

$$\left(\widetilde{A}_j\right)_{\ell, \check{\boldsymbol{n}}} := \frac{1}{\sqrt{m}} T_{\check{\boldsymbol{n}}}(\boldsymbol{x}_{j,\ell}),\ \ell \in [m],\ \check{\boldsymbol{n}} \in [M]^{D-1}, \tag{4.20}$$

so that each one has a restricted isometry constant $\delta_{2s}$ of at most $\delta$ with high probability in its restricted form. Here, the restricted form is introduced by eliminating the columns of the full $\widetilde{A}_j$ indexed by vectors $\boldsymbol{n} \notin \mathcal{I}_{j;\widetilde{n}}$. To explain further, we denote $\widetilde{A}_j P_{j;\widetilde{n}} \boldsymbol{r} = \widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}$ where $P_{j;\widetilde{n}}$ is a restriction matrix defined in (4.11). This comes from the inner product in line 10 of Algorithm 6 which is calculated by using the evaluations of $h_{j,k}$ at $\mathcal{U}_j$ defined in Section 4.1.3. Then, the inner product can be written as $\left(G\widetilde{A}_j P_{j;\widetilde{n}} \boldsymbol{r}\right)_k$ where $G$ is defined as in (4.17). In other words, the evaluations of $h$ at $[u_{j,k}, \boldsymbol{x}_{j,\ell}]$ from $\mathcal{G}_j^I$ in Section 4.1.3 are utilized to compute the inner product. Then, the matrix-vector multiplication $\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}$ can be considered in its restricted form by eliminating the columns of $\widetilde{A}_j$ and elements of $\boldsymbol{r}_{j;\widetilde{n}}$ which are zero due to their corresponding index vectors not belonging to $\mathcal{I}_{j;\widetilde{n}}$. The resulting restricted matrix $\widetilde{A}_j$ has the size $m \times N'$ where $N'$ is bounded above in (4.8) so that $\widetilde{A}_j$ has the restricted isometry constant $\delta_{2s}$ mentioned. The advantage of forming RIP matrices in this fashion is that it allows us to analyze the different iterations of Algorithm 5 repeatedly with the same RIP matrices. For a discussion about the number of measurements needed to ensure that $\widetilde{A}_j$ satisfies the RIP, see the following lemma (which is a simple consequence of Theorem 6).

**Lemma 3.** *Let $\widetilde{A}_j \in \mathbb{C}^{m \times N'}$ be the random sampling matrix as in (4.20) in its restricted form. If, for $\delta, p \in (0,1)$,*

$$m \geq aK^2\delta^{-2}s \max\left\{d \log^2(4s)\log(9m)\log\left(\frac{8e(D-1)DM}{d}\right), \log\left(\frac{D}{p}\right)\right\},$$

*then with probability at least $1-p$, the restricted isometry constant $\delta_s$ of $\widetilde{A}_j$ satisfies $\delta_s \leq \delta$ for all $j \in [D]$. The constant $a > 0$ is universal.*

As we will show, more than half of the proxy functions, $\{h_{j;k}\}_{k \in [L]}$ are guaranteed with high probability to have $\|h_{j;k}\|_{L_2(\mathcal{D}_j,\nu_j)}$ bounded above by $\|\boldsymbol{r}\|_2$ up to some constant, and also $\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right|$ bounded above and below by $\|\boldsymbol{r}_{j;\widetilde{n}}\|_2$ up to some constants for all $\widetilde{n} \in [M]$ and $j \in [D]$.

To show this we consider the indicator variable $E_{h,j,\widetilde{n},k}$ which is 1 if and only if all three of

1. $\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \leq \alpha'\|\boldsymbol{r}\|_2^2$ for the absolute constant $\alpha'$ defined in Lemma 4,
2. $\frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 \geq \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq \frac{\sqrt{23}}{12}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2$, and

3. the vector of Gaussian weights $\boldsymbol{g}^k \in \mathbb{R}^m$ satisfying $\frac{1}{2}m \leq \|\boldsymbol{g}^k\|_2^2 \leq \frac{3}{2}m$,

are simultaneously true, and 0 otherwise.

The proof will proceed as follows. Lemmas 4 and 5 together with the bound on $\|\boldsymbol{g}^k\|_2$ through Bernstein's inequality imply that the probability of each $E_{h,j,\widetilde{n},k}$ being 1 is greater than 0.5. Combining this with Chernoff bound, the deviation of $\sum_{k\in[L]} E_{h,j,\widetilde{n},k}$ below its expectation shows exponential decay in its distribution. Then, with sufficiently many proxy functions, i.e., sufficiently large $L$, the probability that $\sum_{k\in[L]} E_{h,j,\widetilde{n},k} < L/2$ for all $(h, j, \widetilde{n}) \in \mathcal{H} \times [D] \times [M]$ becomes very small, which is shown in Theorem 8. The number $L$ logarithmically depends on $DM|\mathcal{H}|$. In order to get the desired properties for all $2s$-sparse functions satisfying our support assumption, the finite function set $\mathcal{H}$ is taken as $\mathcal{H}^\epsilon$ with corresponding coefficient vector set $\mathcal{R}^\epsilon$ which is an $\epsilon$-cover over all normalized $2s$-sparse vectors in $\mathbb{C}^N$ in Theorem 9. Thus, with high probability, the desired properties hold uniformly, i.e., for all functions of interest, and for $\forall j \in [D]$ and $\widetilde{n} \in [M]$.

The following lemma bounds the energy of the proxy functions.

**Lemma 4.** *Suppose that $\boldsymbol{r} \in \mathbb{C}^N$ is $2s$-sparse, and the restricted isometry constant $\delta_{2s}$ of $\widetilde{A}_j$ satisfies $\delta_{2s} \leq \delta$ for all $j \in [D]$ where $\delta \in (0, 7/16)$. Then, for each $k \in [L]$, there exists an absolute constant $\alpha' \in \mathbb{R}^+$ such that*

$$\mathbb{P}\left[\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \geq \alpha' \|\boldsymbol{r}\|_2^2\right] \leq .025 \tag{4.21}$$

*for all $j \in [D]$.*

*Proof.* Consider the random sampling point set $\mathcal{X}_j = \left\{\boldsymbol{x}_{j,\ell} \mid \ell \in [m]\right\}$ to be fixed for the moment. We begin by noting that

$$
\begin{aligned}
\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} =& \int_{\mathcal{D}_j} |h_{j;k}(x)|^2 d\nu_j(x) \\
=& \int_{\mathcal{D}_j} \left| \frac{1}{\sqrt{m}} \sum_{\ell\in[m]} g^k_\ell h([x, \boldsymbol{x}_{j,\ell}]) \right|^2 d\nu_j(x) \\
=& \frac{1}{m} \sum_{\ell,\ell'\in[m]} g^k_\ell g^k_{\ell'} \int_{\mathcal{D}_j} h([x,\boldsymbol{x}_{j,\ell}])\overline{h([x,\boldsymbol{x}_{j,\ell'}])}\, d\nu_j(x) \\
=& \frac{1}{m} \sum_{\ell,\ell'\in[m]} g^k_\ell g^k_{\ell'} \sum_{\boldsymbol{n},\boldsymbol{n'}\in\mathrm{supp}(\boldsymbol{r})} r_{\boldsymbol{n}}\overline{r_{\boldsymbol{n'}}} \prod_{i\in[D]'} T_{i;n_i}(x_{j,\ell})_i \prod_{i'\in[D]'} \overline{T_{i';n'_{i'}}(x_{j,\ell'})_{i'}} \\
& \times \int_{\mathcal{D}_j} T_{j;n_j}(x)\overline{T_{j;n'_j}(x)}\, d\nu_j(x) \qquad (4.22) \\
=& \frac{1}{m} \sum_{\ell,\ell'\in[m]} g^k_\ell g^k_{\ell'} \left( \sum_{\widetilde{n}\in[M]} \sum_{\substack{\boldsymbol{n},\boldsymbol{n'}\text{ s.t.}\\ n_j=n'_j=\widetilde{n}}} r_{\boldsymbol{n}}\overline{r_{\boldsymbol{n'}}} \prod_{i\in[D]'} T_{i;n_i}(x_{j,\ell})_i \prod_{i'\in[D]'} \overline{T_{i';n'_{i'}}(x_{j,\ell'})_{i'}} \right) \\
=& \sum_{\widetilde{n}\in[M]} \sum_{\ell,\ell'\in[m]} g^k_\ell g^k_{\ell'} \left(\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\right)_\ell \overline{\left(\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\right)_{\ell'}}, \qquad (4.23)
\end{aligned}
$$

where $\widetilde{A}_j \in \mathbb{C}^{m\times N'}$ is the restricted random sampling matrix from (4.20) and $\boldsymbol{r}_{j;\widetilde{n}} \in \mathbb{C}^{N'}$ is the restricted vector from (4.7). Thus, we can see that

$$
\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} = \sum_{\widetilde{n}\in[M]} \left|\langle \widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}, \boldsymbol{g}^k\rangle\right|^2 = \left\|(\boldsymbol{g}^k)^* \widetilde{A}_j R\right\|_2^2 = \left\|\left(\widetilde{A}_j R\right)^* \boldsymbol{g}^k\right\|_2^2
$$

where $R \in \mathbb{C}^{N'\times M}$ is the matrix whose $\widetilde{n}^{\text{th}}$ column is $\boldsymbol{r}_{j;\widetilde{n}}$. This yields results that $\mathbb{E}_{\boldsymbol{g}^k}\left[\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)}\right] = \|\widetilde{A}_j R\|_{\mathrm{F}}^2$.

Now observe that $\left(\widetilde{A}_j R\right)^* \boldsymbol{g}^k \sim \mathcal{N}(\boldsymbol{0}, U\Sigma^2 U^*)$, where $U\Sigma V^*$ is the SVD of $\left(\widetilde{A}_j R\right)^*$ and

$\Sigma \in \mathbb{R}^{\min\{M,m\}\times\min\{M,m\}}$ is the diagonal matrix containing at most $\min\{M,m\}$ nonzero singular values, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{M,m\}} \geq 0$, of $\widetilde{A}_j R$. Let $\widetilde{\boldsymbol{g}}^k := \Sigma V^* \boldsymbol{g}^k \sim \mathcal{N}(\mathbf{0}, \Sigma^2)$ and note that $\|U\widetilde{\boldsymbol{g}}^k\|_2^2 = \|\widetilde{\boldsymbol{g}}^k\|_2^2$. As a consequence, one can see that

$$\mathbb{P}\left[\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \geq t\right] = \mathbb{P}\left[\|\widetilde{\boldsymbol{g}}^k\|_2^2 \geq t\right] = \mathbb{P}\left[\sum_{\ell=1}^{\min\{M,m\}} \sigma_\ell^2 X_\ell \geq t\right]$$

holds for all $t \in \mathbb{R}$, where each $X_\ell$ is an i.i.d $\chi^2$ random variable. Applying the Bernstein type inequality given in Proposition 5.16 in [53] we deduce that

$$\begin{aligned}\mathbb{P}\left[\left|\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} - \|\widetilde{A}_j R\|_{\mathrm{F}}^2\right| \geq t\right] &\leq 2\exp\left(-a' \min\left\{\frac{t^2}{\|\boldsymbol{\sigma}\|_4^4}, \frac{t}{\|\boldsymbol{\sigma}\|_\infty^2}\right\}\right)\\ &\leq 2\exp\left(-a' \min\left\{\frac{t^2}{\|\widetilde{A}_j R\|_F^4}, \frac{t}{\|\widetilde{A}_j R\|_F^2}\right\}\right) \qquad (4.24)\end{aligned}$$

where $a' \in \mathbb{R}^+$ is an absolute constant, and $\boldsymbol{\sigma}$ is the vector containing the diagonal elements of $\Sigma$. An application of (4.24) with $t = \max\left\{\log 80/a', \sqrt{\log 80/a'}\right\} \|\widetilde{A}_j R\|_{\mathrm{F}}^2$ finally tells us that

$$\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \geq \left(1 + \max\left\{\frac{\log 80}{a'}, \sqrt{\frac{\log 80}{a'}}\right\}\right) \|\widetilde{A}_j R\|_{\mathrm{F}}^2$$

will hold with probability at most 1/40.

Turning our attention to $\|\widetilde{A}_j R\|_{\mathrm{F}}^2 = \sum_{\widetilde{n}\in[M]} \|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2^2$, we assert that

$$\|\widetilde{A}_j R\|_{\mathrm{F}}^2 = \sum_{\widetilde{n}\in[M]} \|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2^2 \geq \frac{1}{2} \sum_{\widetilde{n}\in[M]} \|\boldsymbol{r}_{j;\widetilde{n}}\|_2^2 = \frac{1}{2}\|\boldsymbol{r}\|_2^2$$

holds since the restricted isometry constant $\delta_{2s}$ of $\widetilde{A}_j$ is assumed to be bounded above by $\frac{7}{16}$. Therefore, we finally get the desired probability estimate with $\alpha' := \frac{1}{2}\big(1 + \max\big\{\frac{\log 80}{a'},$

$\sqrt{\frac{\log 80}{a'}}\}).$ □

The following lemma bounds the estimated inner products.

**Lemma 5.** *Suppose that $\boldsymbol{r} \in \mathbb{C}^N$ is $2s$-sparse, and the restricted isometry constant $\delta_{2s}$ of $\widetilde{A}_j$ satisfies $\delta_{2s} \leq \delta$ for all $j \in [D]$ where $\delta \in (0, 7/16)$. Let $k \in [L]$, $j \in [D]$, and $\widetilde{n} \in [M]$. Then, there exists an absolute constant $\beta' \in \mathbb{R}^+$ such that*

$$\mathbb{P}\left[\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \leq \frac{\sqrt{23}}{12}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 \text{ or } \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq \frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2\right] \leq 0.273. \quad (4.25)$$

*Proof.* Consider the random sampling point set $\mathcal{X}_j$ to be fixed for the time being. Recalling the definitions of $h_{j;k}$ and of $\boldsymbol{r}_{j;\widetilde{n}}$, one can see that

$$\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)} = \frac{1}{\sqrt{m}} \sum_{\ell \in [m]} g_\ell^k \sum_{\substack{\boldsymbol{n} \text{ s.t.} \\ n_j = \widetilde{n}}} r_{\boldsymbol{n}} \prod_{i \in [D]'} T_{i;n_i}(x_{j,\ell})_i = \sum_{\ell \in [m]} g_\ell^k \left(\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\right)_\ell. \quad (4.26)$$

Looking at (4.26) one can see that $\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)} \sim \mathcal{N}(0, \|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2^2)$ and hence,

$$\mathbb{P}\left[\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \leq \frac{\|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2}{3} \text{ or } \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq 3\|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2\right] \leq 0.273 \quad (4.27)$$

holds. Combining (4.27) and the assumption on $\delta_{2s}$, which yields $\frac{9}{16}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2^2 \leq \|\widetilde{A}_j \boldsymbol{r}_{j;\widetilde{n}}\|_2^2 \leq \frac{23}{16}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2^2$, establishes the desired result. □

**Theorem 8.** *Let $\mathcal{H}$ be a finite set of functions $h$ whose BOS coefficient vectors are $2s$-sparse, and let $\boldsymbol{r}_h \in \mathbb{C}^N$ denote the coefficient vector for each $h \in \mathcal{H}$. Suppose that the restricted isometry constant $\delta_{2s}$ of $\widetilde{A}_j$ satisfies $\delta_{2s} \leq \delta$ for all $j \in [D]$ where $\delta \in (0, 7/16)$. Furthermore,* *let $p \in (0,1)$, $L \in \mathbb{N}$ be odd, and $L \geq \widetilde{\gamma} \log(DM|\mathcal{H}|/p)$ hold for a sufficiently large absolute constant $\widetilde{\gamma} \in \mathbb{R}^+$. Then, $\sum_{k \in [L]} E_{h,j,\widetilde{n},k} > L/2$ simultaneously for all $(h,j,\widetilde{n}) \in \mathcal{H} \times [D] \times [M]$ with probability at least $1-p$. That is, with probability at least $1-p$, the following will hold simultaneously for each $(h,j,\widetilde{n}) \in \mathcal{H} \times [D] \times [M]$: All three of*

1. *$\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \leq \alpha' \|\boldsymbol{r}_h\|_2^2$ for the absolute constant $\alpha'$ defined in Lemma 4,*
2. *$\frac{9}{4}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2 \geq \left|\langle h_{j;k}, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq \frac{\sqrt{23}}{12}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2$, and*
3. *the vector of Gaussian weights $\boldsymbol{g}^k \in \mathbb{R}^m$ satisfying $\frac{1}{2}m \leq \|\boldsymbol{g}^k\|_2^2 \leq \frac{3}{2}m$,*

*will be simultaneously true for more than half of the $k \in [L]$.*

*Proof.* Let $h \in \mathcal{H}$, $k \in [L]$, and $\widetilde{n} \in [M]$. The probabilities that the first and second properties fail are given in (4.21) and (4.25), respectively. For the third property, applying the Bernstein type inequality given in Proposition 5.16 in [53], one obtains

$$\mathbb{P}\left[\left|\|\boldsymbol{g}^k\|_2^2 - m\right| \geq \frac{m}{2}\right] \leq 2\mathrm{e}^{-a''m} \leq 0.03, \tag{4.28}$$

where $a'' \in \mathbb{R}^+$ is an absolute constant.

Combining (4.21), (4.25), (4.28) via a union bound now tell us that $\mathbb{P}\left[E_{h,j,\widetilde{n},k} = 0\right] \leq 328/1000$. Utilizing the Chernoff bound (see, e.g., [54, 55]) one now sees that

$$\mathbb{P}\left[\sum_{k\in[L]} E_{h,j,\widetilde{n},k} < L/2\right] = \mathbb{P}\left[\sum_{k\in[L]} (1 - E_{h,j,\widetilde{n},k}) > L/2\right] < \mathrm{e}^{-L/\bar{\gamma}} \leq \frac{p}{DM|\mathcal{H}|}$$

for an absolute constant $\bar{\gamma} \in \mathbb{R}^+$, where the last inequality follows by choosing $\widetilde{\gamma} = \bar{\gamma}$ in the assumption. Applying the union bound over all choices of $(h,j,\widetilde{n}) \in \mathcal{H} \times [D] \times [M]$ now establishes the desired result. □

**Theorem 9.** *Let $\mathcal{H}_{2s}$ be the set of all functions $h$ whose coefficient vectors are $2s$-sparse,* *and let* $\boldsymbol{r}_h \in \mathbb{C}^N$ *denote the coefficient vector for each* $h \in \mathcal{H}_{2s}$*. Suppose that the restricted isometry constant* $\delta_{2s}$ *of* $\widetilde{A}_j$ *satisfies* $\delta_{2s} \leq \delta$ *for all* $j \in [D]$ *where* $\delta \in (0, 7/16)$*. Furthermore, let* $p \in (0,1)$*,* $L \in \mathbb{N}$ *be odd, and assume that* $L \geq \gamma' s \cdot d \cdot \log\left(\frac{DM}{\sqrt[d]{sd\ p^{1/s}}}\right)$ *for sufficiently large absolute constant* $\gamma' \in \mathbb{R}^+$*. Then, with probability greater than* $1-p$*, one has* $\sum_{k\in[L]} E_{h,j,\widetilde{n},k} > L/2$ *simultaneously for all* $(h, j, \widetilde{n}) \in \mathcal{H} \times [D] \times [M]$*. Consequently, with probability greater than* $1-p$*, it will hold that for all choices of* $(h, j, \widetilde{n}) \in \mathcal{H}_{2s} \times [D] \times [M]$ *both*

1. $\|h_{j;k}\|^2_{L^2(\mathcal{D}_j,\nu_j)} \leq (\alpha'+1)\|\boldsymbol{r}_h\|_2^2$ *for the absolute constant* $\alpha'$ *defined in Lemma 4, and*
2. $\frac{9}{2}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2 \geq \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq \frac{1}{3}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2$

*are true simultaneously for more than half of the* $k \in [L]$*.*

*Proof.* Define $\mathcal{R}^\epsilon \subset \mathbb{C}^N$ as a finite $\epsilon$-cover of all $2s$-sparse coefficient vectors $\boldsymbol{r} \in \mathbb{C}^N$ with $\|\boldsymbol{r}\|_2 = 1$, together with $\mathbf{0}$, where $N = \binom{D}{d}M^d$ and $\epsilon \in (0,1)$. Such covers exist of cardinality $|\mathcal{R}^\epsilon| \leq \left(\frac{eN}{2s}\right)^{2s}\left(1+\frac{2}{\epsilon}\right)^{2s}$ (see, e.g., Appendix C of [37]). Define $\mathcal{H}^\epsilon$ as the set of functions corresponding to the $2s$-sparse coefficient vectors in $\mathcal{R}^\epsilon$. Assume that for $\mathcal{H} = \mathcal{H}^\epsilon$ and all choices of $(h, j, \widetilde{n}) \in \mathcal{H}^\epsilon \times [D] \times [M]$, Properties 1 – 3 of Theorem 8 will hold for more than half of the $k \in [L]$. By the theorem, this event will happen with probability at least $1-p$. We will now prove that under this assumption both Properties 1 and 2 above will hold as desired.

Let $\widetilde{n} \in [M]$, $j \in [D]$, consider $h \in \mathcal{H}_{2s}$ with coefficient vector $\boldsymbol{r} = \boldsymbol{r}_h$, and let $\tau := \|\boldsymbol{r}\|_2^2$. Then, there exists an $h' \in \mathcal{H}^\epsilon$ with coefficient vector $\boldsymbol{r}' \in \mathcal{R}^\epsilon$ such that both $\|\boldsymbol{r}'\|_2 = 1$ and $\left\|\boldsymbol{r} - \tau\boldsymbol{r}'\right\|_2 \leq \epsilon\tau$ hold. Finally, let $k \in [L]$ be one of the values for which Properties 1 – 3 of Theorem 8 are simultaneously true for $(h', j, \widetilde{n})$. We will begin by establishing Property 1 above for $h$, $j$ and $k$. Using (4.19) we have that

$$\begin{aligned}
\left\|h_{j;k}\right\|_{L^2(\mathcal{D}_j,\nu_j)} &= \left\| \frac{1}{\sqrt{m}} \sum_{\ell\in[m]} g_\ell^k \sum_{\boldsymbol{n}\in\operatorname{supp}(\boldsymbol{r})} r_{\boldsymbol{n}} T_{j;n_j}(x) \prod_{i\in[D]'} T_{i;n_i}(x_{j,\ell})_i \right\|_{L^2(\mathcal{D}_j,\nu_j)} \\
&\leq \tau \left\|h'_{j;k}\right\|_{L^2(\mathcal{D}_j,\nu_j)} \qquad (4.29)\\
&+ \left\| \frac{1}{\sqrt{m}} \sum_{\ell\in[m]} g_\ell^k \sum_{\boldsymbol{n}\in\operatorname{supp}(\boldsymbol{r})} \left(r_{\boldsymbol{n}} - \tau r'_{\boldsymbol{n}}\right) T_{j;n_j}(x) \prod_{i\in[D]'} T_{i;n_i}(x_{j,\ell})_i \right\|_{L^2(\mathcal{D}_j,\nu_j)} \\
&\leq \tau\sqrt{\alpha'}\|\boldsymbol{r}'\|_2 + \left\|\left(h - \tau h'\right)_{j;k}\right\|_{L^2(\mathcal{D}_j,\nu_j)} \\
&= \tau\sqrt{\alpha'} + \left\|\left(h - \tau h'\right)_{j;k}\right\|_{L^2(\mathcal{D}_j,\nu_j)}, \qquad (4.30)
\end{aligned}$$

where the last inequality follows from the first property of Theorem 8 holding for $h'$.

Repeating the expansion from the proof of Lemma 4 for $\left\|\left(h - \tau h'\right)_{j;k}\right\|^2_{L^2(\mathcal{D}_j,\nu_j)}$, one obtains

$$\begin{aligned}
\left\|\left(h - \tau h'\right)_{j;k}\right\|^2_{L^2(\mathcal{D}_j,\nu_j)} &= \sum_{\widetilde{n}\in[M]} \left| \left\langle \widetilde{A}_j \left(\boldsymbol{r} - \tau\boldsymbol{r}'\right)_{j;\widetilde{n}}, \boldsymbol{g}^k \right\rangle \right|^2 \\
&\leq \sum_{\widetilde{n}\in[M]} \left\|\widetilde{A}_j \left(\boldsymbol{r} - \tau\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2^2 \left\|\boldsymbol{g}^k\right\|_2^2 \\
&\leq \frac{9}{4} m \sum_{\widetilde{n}\in[M]} \left\|\left(\boldsymbol{r} - \tau\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2^2
\end{aligned}$$

where the last inequality follows from the third property of Theorem 8, and $\widetilde{A}_j$ having $\delta_{2s} \leq \frac{7}{16}$. Continuing, we can see that

$$\left\|\left(h - \tau h'\right)_{j;k}\right\|^2_{L^2(\mathcal{D}_j,\nu_j)} \leq \frac{9}{4} m \left\|\left(\boldsymbol{r} - \tau\boldsymbol{r}'\right)\right\|_2^2 \leq \frac{9}{4} m\tau^2\epsilon^2.$$

Combining this expression with (4.30) we now learn that

$$\left\|h_{j;k}\right\|_{L^2(\mathcal{D}_j,\nu_j)} \leq \tau\left(\sqrt{\alpha'} + \frac{3}{2}\epsilon\sqrt{m}\right) = \|\boldsymbol{r}\|_2\left(\sqrt{\alpha'} + \frac{3}{2}\epsilon\sqrt{m}\right).$$

Making sure to use, e.g., an $\epsilon \leq \left(6\sqrt{\alpha' m}\right)^{-1}$ ensures property one.

Turning our attention to establishing Property 2 above for $h$, $\widetilde{n}$, and $k$, we now choose an $h' \in \mathcal{H}^\epsilon$ whose coefficient vector $\boldsymbol{r}' \in \mathcal{R}^\epsilon$ has $\|\boldsymbol{r}'\|_2 = 1$, and also satisfies

$$\left\|\boldsymbol{r}_{j;\widetilde{n}} - \tau'\boldsymbol{r}'\right\|_2 \leq \epsilon\tau' \tag{4.31}$$

for $\tau' := \|\boldsymbol{r}_{j;\widetilde{n}}\|_2$. Note that all nonzero entries of $\boldsymbol{r}'_{j;\widetilde{n}}$ agree with those of $\boldsymbol{r}'$, the latter vector only has certain additional nonzero entries in locations where $\boldsymbol{r}'_{j;\widetilde{n}}$ and also $\boldsymbol{r}_{j;\widetilde{n}}$ vanish. Consequently, replacing $\boldsymbol{r}'$ by $\boldsymbol{r}'_{j;\widetilde{n}}$ makes the left hand side of (4.31) smaller, and one obtains that

$$\left\|\boldsymbol{r}_{j;\widetilde{n}} - \tau'\boldsymbol{r}'_{j;\widetilde{n}}.\right\|_2 \leq \epsilon\tau' \tag{4.32}$$

From (4.19) one can see that

$$\begin{aligned}\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| &\geq \left|\langle \tau' h'_k, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| - \left|\left\langle\left(h_{j;k} - \tau' h'_k\right), T_{j;\widetilde{n}}\right\rangle_{(\mathcal{D}_j,\nu_j)}\right| \\ &\geq \frac{\sqrt{23}}{12}\tau'\|\boldsymbol{r}'_{j;\widetilde{n}}\|_2 - \left|\sum_{\ell\in[m]} g^k_\ell\left(\widetilde{A}_j\left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right)_\ell\right|.\end{aligned}$$

where the last inequality follows from the second property of Theorem 8 holding for $h'$. Continuing using (4.32) we have that

$$\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| \geq \frac{\sqrt{23}}{12}\left(\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 - \|(\tau'\boldsymbol{r}' - \boldsymbol{r})_{j;\widetilde{n}}\|_2\right) - \left|\left\langle\widetilde{A}_j\left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}, \boldsymbol{g}^k\right\rangle\right|$$

$$
\begin{aligned}
&\geq \frac{\sqrt{23}}{12}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 - \frac{\sqrt{23}}{12}\epsilon\tau' - \left|\left\langle \widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}, \boldsymbol{g}^k \right\rangle\right| \\
&\geq \frac{\sqrt{23}}{12}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 - \frac{\sqrt{23}}{12}\epsilon\tau' - \left\|\widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2 \left\|\boldsymbol{g}^k\right\|_2 \\
&\geq \frac{\sqrt{23}}{12}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 - \frac{\sqrt{23}}{12}\epsilon\tau' - \frac{3}{2}\sqrt{m}\left\|\left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2 \\
&= \frac{1}{3}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 \left(\frac{\sqrt{23}}{4} - \frac{\sqrt{23}\epsilon}{4} - \frac{9}{2}\epsilon\sqrt{m}\right).
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| &\leq \left|\langle \tau' h'_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| + \left|\left\langle \left(h_{j;k} - \tau' h'_{j;k}\right), T_{j;\widetilde{n}}\right\rangle_{(\mathcal{D}_j,\nu_j)}\right| \\
&\leq \frac{9}{4}\tau'\|\boldsymbol{r}'_{j;\widetilde{n}}\|_2 + \left|\sum_{\ell\in[m]} g_\ell^k \left(\widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right)_\ell\right|.
\end{aligned}
$$

As above, we obtain using (4.32) that

$$
\begin{aligned}
\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j,\nu_j)}\right| &\leq \frac{9}{4}\left(\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 + \|(\tau'\boldsymbol{r}' - \boldsymbol{r})_{j;\widetilde{n}}\|_2\right) + \left|\left\langle \widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}, \boldsymbol{g}^k \right\rangle\right| \\
&\leq \frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 + \frac{9}{4}\epsilon\tau' + \left|\left\langle \widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}, \boldsymbol{g}^k \right\rangle\right| \\
&\leq \frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 + \frac{9}{4}\epsilon\tau' + \left\|\widetilde{A}_j \left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2 \left\|\boldsymbol{g}^k\right\|_2 \\
&\leq \frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 + \frac{9}{4}\epsilon\tau' + \frac{3}{2}\sqrt{m}\left\|\left(\boldsymbol{r} - \tau'\boldsymbol{r}'\right)_{j;\widetilde{n}}\right\|_2 \\
&= \frac{9}{4}\|\boldsymbol{r}_{j;\widetilde{n}}\|_2 \left(1 + \epsilon + \frac{2}{3}\epsilon\sqrt{m}\right).
\end{aligned}
$$

Once again, making sure to use, e.g., an $\epsilon \leq \left(6\sqrt{\alpha' m}\right)^{-1}$ will now ensure property two for $h$ as well. □

**Lemma 6.** *Let $\mathcal{H}_{2s}$ be the set of all functions $h$ whose coefficient vectors are $2s$-sparse, and let $\boldsymbol{r}_h \in \mathbb{C}^N$ denote the coefficient vector for each $h \in \mathcal{H}_{2s}$. Furthermore, let $\delta \in$*

*$(0, 7/16)$, $p \in (0,1)$, $L \in \mathbb{N}$ be odd, and assume that $m \geq \widetilde{\beta}' K^2 \delta^{-2} s \max\left\{ d \log^2(4s) \log(9m) \log\left(\frac{8\mathrm{e}(D-1)DM}{d}\right), \log\left(\frac{2D}{p}\right)\right\}$ and $L \geq \gamma' s \cdot d \cdot \log\left(\frac{DM}{\sqrt[d]{sd\ (p/2)^{1/s}}}\right)$ for sufficiently large absolute constants $\widetilde{\beta}', \gamma' \in \mathbb{R}^+$. Then, with probability greater than $1-p$, all of the following will hold for all $(h, j, \widetilde{n}) \in \mathcal{H}_{2s} \times [D] \times [M]$: Both*

1. *$\|h_{j;k}\|^2_{L^2(\mathcal{D}_j, \nu_j)} \leq (\alpha' + 1)\|\boldsymbol{r}_h\|_2^2$ for the absolute constant $\alpha'$ defined in Lemma 4, and*
2. *$\frac{9}{2}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2 \geq \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j, \nu_j)}\right| \geq \frac{1}{3}\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\|_2$*

*will be simultaneously true for more than half of the $k \in [L]$.*

*Proof.* Let $A$ be the event that for all $(h, j, \widetilde{n}) \in \mathcal{H}_{2s} \times [D] \times [M]$, the properties 1 and 2 in Theorem 9 simultaneously hold for more than half of the $k \in [L]$, and let $B$ be the event that the restricted isometry constant $\delta_{2s}$ of $\widetilde{A}_j$ satisfies $\delta_{2s} \leq \delta$ for all $j \in [D]$. By Theorem 9 and Lemma 3 with properly chosen parameters including $L$ and $m$, both $\mathbb{P}\left[A \mid B\right]$ and $\mathbb{P}[B]$ are greater than $1 - p/2$. We obtain, by Bayes' theorem,

$$1 - p \leq \left(1 - \frac{p}{2}\right)\left(1 - \frac{p}{2}\right) \leq \mathbb{P}\left[A \mid B\right]\mathbb{P}\left[B\right] = \mathbb{P}[A \cap B] \leq P[A],$$

which establishes the desired result. $\square$

The results from Lemma 6 can be exploited in our algorithm as follows. In the entry identification, by taking the median over $k \in [L]$ of $\left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j, \nu_j)}\right|$, we get a nonzero value if $\left\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\right\|_2$ is nonzero and a zero value if $\left\|(\boldsymbol{r}_h)_{j;\widetilde{n}}\right\|_2$ is zero, especially due to the second property in Lemma 6 being satisfied for more the half of $k \in [L]$. Thus, we store all those $\widetilde{n}$ with nonzero median value in $\mathcal{N}_j$. On the other hand, the summation over $\widetilde{n} \in [M]$ of $\mathrm{median}_k \left|\langle h_{j;k}, T_{j;\widetilde{n}}\rangle_{(\mathcal{D}_j, \nu_j)}\right|$ can be also used for the halting criterion in our algorithm. Although $\mathcal{O}(\|\boldsymbol{c}\|_2/\eta)$ iterations guarantee the desired precision, it is not necessary to repeat the iteration if the residual $\boldsymbol{r}$ already has small energy. For any $j \in [D]$, if

$$\sum_{\widetilde{n} \in [M]} \left| \mathrm{median}_k |\langle h_{j;k}, T_{j;\widetilde{n}} \rangle_{(\mathcal{D}_j, \nu_j)}| \right|^2 \leq \frac{1}{9}\eta^2,$$

then $\|\boldsymbol{r}\|_2 \leq \eta$ by using the lower bound in the Property 2 of Lemma 6.

## 4.3.2 Pairing

In the entry identification, we can find at most $2s$ entries belonging to $\mathcal{N}_j := \{n_j \mid \boldsymbol{n} \in \mathrm{supp}(\boldsymbol{r})\} \subset [M]$ for each $j \in [D]$. However, we do not know how to combine the entries of each $\mathcal{N}_j$ to identify the elements of $\mathrm{supp}(\boldsymbol{r})$. In order to do this, in the pairing process briefly introduced in Section 4.2.2, we successively build up the prefix set $\mathcal{P}_j$ of the energetic pairs for all $j \in [D] \setminus \{0\}$ such that $\mathcal{P}_j \supset \left\{ \widetilde{\boldsymbol{n}} \in \mathcal{P}_{j-1} \times \mathcal{N}_j \mid \|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$ with the initialization of $\mathcal{P}_0 = \mathcal{N}_0$. The prefix set $\mathcal{P}_j$ contains only $2s$ pairs throwing out the other pairs with smaller energy for each $j \in [D] \setminus \{0\}$ so that $\mathcal{P}_{D-1} \supset \left\{ \widetilde{\boldsymbol{n}} \in \mathrm{supp}(\boldsymbol{r}) \mid |r_{\widetilde{\boldsymbol{n}}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$. From (4.13), the energy $\left\| \boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)} \right\|_2^2$ corresponding to each $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$ has the following equality,

$$\left\| \boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)} \right\|_2^2 = \left\| \langle h, T_{j;\widetilde{\boldsymbol{n}}} \rangle_{L^2\left(\times_{i \in [j+1]} \mathcal{D}_i, \otimes_{i \in [j+1]} \nu_i\right)} \right\|_{L^2\left(\mathcal{D}_j'', \boldsymbol{\nu}_j''\right)}^2 . \tag{4.33}$$

The energy is estimated by using the following estimator $E_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ defined as

$$E_{j;(\widetilde{\boldsymbol{n}},\cdots)} := \frac{1}{m_2} \sum_{k \in [m_2]} \left| \frac{1}{m_1} \sum_{\ell \in [m_1]} h(\boldsymbol{w}_{j,\ell}, \boldsymbol{z}_{j,k}) \overline{T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_{j,\ell})} \right|^2$$

which approximates the right hand side of (4.33) by using only a finite evaluations of $h$. Those sampling point sets $\mathcal{W}_j \times \mathcal{Z}_j$ for all $j \in [D] \setminus \{0\}$ are constructed from $\mathcal{W}_j := \{\boldsymbol{w}_{j,\ell}\}_{\ell \in [m_1]}$ and

$\mathcal{Z}_j := \{\boldsymbol{z}_{j,k}\}_{k\in[m_2]}$ where $\boldsymbol{w}_{j,\ell}$ and $\boldsymbol{z}_{j,k}$ are chosen independently at random from $\times_{i\in[j+1]}\mathcal{D}_i$ and $\mathcal{D}''_j$ for $j \in [D-1]\setminus\{0\}$, respectively. If $j = D-1$, $\mathcal{W}_{D-1} := \{\boldsymbol{w}_{D-1,\ell}\}_{\ell\in[m_1]}$ is chosen from $\times_{i\in[D]}\mathcal{D}_i$ and $\mathcal{Z}_{D-1} = \emptyset$. Note that $\mathcal{W}_j \times \mathcal{Z}_j = \mathcal{G}^P_j$ from Section 4.1.3. Furthermore, the sets $\mathcal{W}_j \times \mathcal{Z}_j$ for all $j \in [D]\setminus\{0\}$ build a random sampling matrix $A^P$ in (4.16) which explicitly expresses the samples(evaluations) of the $2s$-sparse $h$ used in the (4.34) as $A^P\boldsymbol{r}$. The matrix $A^P$ is broken into smaller matrices $B_j$ and $C_j$ for $j \in [D]\setminus\{0\}$ defined and explained in the next paragraph for the complete analysis of the pairing process in the upcoming lemmas and theorems in this section.

For all $j \in [D-1]\setminus\{0\}$, the measurement matrix $C_j \in \mathbb{C}^{m_2\times[M]^{D-j-1}}$ is defined as

$$\left(C_j\right)_{k,\boldsymbol{n}_2} := \frac{1}{\sqrt{m_2}} T''_{\boldsymbol{n}_2}(\boldsymbol{z}_{j,k}), \quad k \in [m_2],\ \boldsymbol{n}_2 \in [M]^{D-j-1}, \tag{4.34}$$

where $T''_{\boldsymbol{n}_2}(\boldsymbol{y})$ is a partial product of the last $D-j-1$ terms of $T_{\boldsymbol{n}}(\boldsymbol{x})$ defined in (4.1), i.e.,

$$T''_{\boldsymbol{n}_2}(\boldsymbol{y}) = \prod_{i\in[D-j-1]} T_{i+j+1;n_{i+j+1}}(y_i), \quad \boldsymbol{y} \in \mathcal{D}''_j.$$

The matrix $C_j$ can be restricted to the matrix of size $m_2 \times \widetilde{N}_j$ when $C_j$ is applied to $\boldsymbol{v}_{j,(\widetilde{n},\cdots)}$ where $\widetilde{N}_j$ estimated in (4.10) is the cardinality of the superset of any possible $\operatorname{supp}(\boldsymbol{v}_{j;\widetilde{\boldsymbol{n}}})$ with fixed $j, \widetilde{\boldsymbol{n}}, D$ and $d$ so that it satisfies RIP with sufficiently large $m_2$. When the $j = D-1$, the matrix $C_j$ is defined to be 1 since $\mathcal{Z}_{D-1} = \emptyset$. For all $j \in [D]\setminus\{0\}$, on the other hand, the matrices $B_j \in \mathbb{C}^{m_1\times[M]^{j+1}}$ is defined as

$$\left(B_j\right)_{\ell,\boldsymbol{n}_1} := \frac{1}{\sqrt{m_1}} T'_{\boldsymbol{n}_1}(\boldsymbol{w}_{j,\ell}), \quad \ell \in [m_1],\ \boldsymbol{n}_1 \in [M]^{j+1}, \tag{4.35}$$

where $T'_{\boldsymbol{n}_1}(\boldsymbol{z})$ is a partial product of the first $j+1$ terms of $T_{\boldsymbol{n}}(\boldsymbol{x})$ in (4.1), i.e.,

$$T'_{\boldsymbol{n}_1}(\boldsymbol{z}) = \prod_{i\in[j+1]} T_{i;n_i}(z_i), \quad \boldsymbol{z} \in \times_{i\in[j+1]}\mathcal{D}_i.$$

The matrix $B_j$ can be restricted to the matrix of size $m_1 \times \bar{N}_j$ where $\bar{N}_j = \binom{j+1}{d}M^d$ if $j+1 \geq d$, or $M^d$ otherwise. The number $\bar{N}_j$ is the cardinality of the set of any possible prefix $\widetilde{n} \in [M]^{j+1}$ with fixed $j, \widetilde{n}$ and $d$. The sampling numbers $m_1$ and $m_2$ are chosen for all $B_j$ and $C_j$ with any $j \in [D] \setminus \{0\}$ to satisfy RIP with the restricted isometry constants $\widetilde{\delta}_{2s+1} \leq \widetilde{\delta}$ and $\delta'_{2s} \leq \delta'$, respectively. We again mention that $C_{D-1} = 1$.

**Lemma 7.** *Let $\mathcal{H}_{2s}$ be the set of all functions $h$ whose coefficient vectors are $2s$-sparse and let $\boldsymbol{r}_h \in \mathbb{C}^N$ denote the coefficient vector for each $h \in \mathcal{H}_{2s}$. Let $j \in [D] \setminus \{0\}$, and $\widetilde{\delta}$ and $\delta'$ be chosen from $(0,1)$. Assume that $B_j$ and $C_j$ satisfy RIP with $\widetilde{\delta}_{2s+1} \leq \widetilde{\delta}$ and $\delta'_{2s} \leq \delta'$, respectively. Then, denoting $\boldsymbol{r} := \boldsymbol{r}_h$ for simplicity,*

$$\begin{aligned}(1-\delta'_{2s})\left(\max\{0, \left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 - \widetilde{\delta}_{2s+1}\|\boldsymbol{r}\|_2\}\right)^2 &\leq E_{j;(\widetilde{\boldsymbol{n}},\cdots)} \\ &\leq (1+\delta'_{2s})\left(\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 + \widetilde{\delta}_{2s+1}\|\boldsymbol{r}\|_2\right)^2.\end{aligned} \tag{4.36}$$

*for any $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$.*

*Proof.* As $j \in [D]\setminus\{0\}$ is fixed, for simplicity, we use notation $\boldsymbol{w}_\ell$ and $\boldsymbol{z}_k$ for sampling points instead of $\boldsymbol{w}_{j,\ell}$ and $\boldsymbol{z}_{j,k}$ constructing the sampling matrices $B_j$ and $C_j$ as in (4.35) and (4.34), respectively. Fix $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$. Letting $\boldsymbol{n} = (\boldsymbol{n}_1, \boldsymbol{n}_2)$, $\boldsymbol{n}_1 \in [M]^{j+1}$ and $\boldsymbol{n}_2 \in [M]^{D-j-1}$, we can rewrite the energy estimate $E_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ as follows,

$$\begin{aligned}
E_{j;(\widetilde{\boldsymbol{n}},\cdots)} &= \frac{1}{m_2}\sum_{k\in[m_2]}\left|\frac{1}{m_1}\sum_{\ell\in[m_1]} h(\boldsymbol{w}_\ell,\boldsymbol{z}_k)\overline{T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_\ell)}\right|^2 \\
&= \frac{1}{m_2}\sum_{k\in[m_2]}\left|\frac{1}{m_1}\sum_{\ell\in[m_1]}\sum_{\boldsymbol{n}=(\boldsymbol{n}_1,\boldsymbol{n}_2)\in\operatorname{supp}(\boldsymbol{r})} r_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{w}_\ell,\boldsymbol{z}_k)\overline{T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_\ell)}\right|^2 \\
&= \frac{1}{m_2}\sum_{k\in[m_2]}\frac{1}{m_1^2}\left|\sum_{\boldsymbol{n}\in\operatorname{supp}(\boldsymbol{r})}\sum_{\ell\in[m_1]} r_{\boldsymbol{n}} T'_{\boldsymbol{n}_1}(\boldsymbol{w}_\ell)\overline{T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_\ell)}T''_{\boldsymbol{n}_2}(\boldsymbol{z}_k)\right|^2 \\
&=: \frac{1}{m_2}\sum_{k\in[m_2]}\frac{1}{m_1^2}\left|\sum_{\substack{\boldsymbol{n}_2 \text{ s.t. } \exists \boldsymbol{n}_1 \\ \text{with } (\boldsymbol{n}_1,\boldsymbol{n}_2)\in\operatorname{supp}(\boldsymbol{r})}} (\widetilde{r}_{\widetilde{\boldsymbol{n}}})_{\boldsymbol{n}_2} T''_{\boldsymbol{n}_2}(\boldsymbol{z}_k)\right|^2, \qquad (4.37)
\end{aligned}$$

where

$$(\widetilde{r}_{\widetilde{\boldsymbol{n}}})_{\boldsymbol{n}_2} := \sum_{\substack{\boldsymbol{n}'_1 \text{ s.t.} \\ \boldsymbol{n}=(\boldsymbol{n}_1,\boldsymbol{n}_2)\in\operatorname{supp}(\boldsymbol{r})}} r_{\boldsymbol{n}} \sum_{\ell\in[m_1]} T'_{\boldsymbol{n}_1}(\boldsymbol{w}_\ell)\overline{T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_\ell)}. \qquad (4.38)$$

We can construct a vector $\widetilde{\boldsymbol{r}} := \left((\widetilde{r}_{\widetilde{\boldsymbol{n}}})_{\boldsymbol{n}_2}\right) \in \mathbb{C}^{\widetilde{N}_j}$ with entries $(\widetilde{r}_{\widetilde{\boldsymbol{n}}})_{\boldsymbol{n}_2}$ at $\boldsymbol{n}_2$ so that $\widetilde{\boldsymbol{r}}$ has a support whose cardinality is at most $2s$ since $h$ is $2s$-sparse. Thus, the energy estimate $E_{j;(\widetilde{\boldsymbol{n}},\cdots)}$ in (4.37) can be expressed as $\left\|C_j\left(\frac{\widetilde{\boldsymbol{r}}}{m_1}\right)\right\|_2^2$. Since the restricted measurement matrix $C_j \in \mathbb{C}^{m_2\times\widetilde{N}_j}$ satisfies the RIP,

$$(1-\delta'_{2s})\left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2^2 \le \left\|C_j\left(\frac{\widetilde{\boldsymbol{r}}}{m_1}\right)\right\|_2^2 \le (1+\delta'_{2s})\left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2^2. \qquad (4.39)$$

In order to get an upper bound and lower bound of $\left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2$, we define $\boldsymbol{e}_{\widetilde{\boldsymbol{n}}} \in \mathbb{C}^{\bar{N}_j}$ as a standard basis vector with all 0 entries except for a 1 at $\widetilde{\boldsymbol{n}}$, and $R_j \in \mathbb{C}^{\widetilde{N}_j \times \bar{N}_j}$ as

$$(R_j)_{\boldsymbol{n}_2,\boldsymbol{n}_1} := \begin{cases} r_{(\boldsymbol{n}_1,\boldsymbol{n}_2)} & \text{if } (\boldsymbol{n}_1,\boldsymbol{n}_2) \in \operatorname{supp}(\boldsymbol{r}) \\ 0 & \text{otherwise} . \end{cases}$$

Note that $R_j$ contains at most $2s$ nonzero elements since $h$ is $2s$-sparse, and $\widetilde{\boldsymbol{n}}$ is any element in $[M]^{j+1}$. Set $\mathcal{Q} := \{\widetilde{\boldsymbol{n}}\} \cup \{\boldsymbol{n}_1 \in [M]^{j+1} \mid \exists \boldsymbol{n}_2 \in [M]^{D-j-1} \text{ such that } (\boldsymbol{n}_1,\boldsymbol{n}_2) \in \operatorname{supp}(\boldsymbol{r})\}$ with a fixed $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$. Thus, the cardinality of $\mathcal{Q}$ is at most $2s+1$. Orderings of indices $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ depend on the column orderings of $B_j$ and $C_j$, respectively. We note that the $\widetilde{\boldsymbol{n}}$-th column of $R_j$ is $\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}$. Both bounds of $\left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2$ are found as follows

$$\begin{aligned}
\left\|\frac{\widetilde{\boldsymbol{r}}}{m_1} - \boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 &= \left\|R_j (B_j)^*_{\mathcal{Q}} (B_j)_{\mathcal{Q}} \boldsymbol{e}_{\widetilde{\boldsymbol{n}}} - R_j \boldsymbol{e}_{\widetilde{\boldsymbol{n}}}\right\|_2 \\
&\le \|R_j\|_{2\to 2} \|(B_j)^*_{\mathcal{Q}}(B_j)_{\mathcal{Q}} - I\|_{2\to 2} \|\boldsymbol{e}_{\widetilde{\boldsymbol{n}}}\|_2 \\
&\le \widetilde{\delta}_{2s+1} \|R_j\|_F \\
&= \widetilde{\delta}_{2s+1} \|\boldsymbol{r}\|_2
\end{aligned}$$

and therefore,

$$\begin{aligned}
&\left| \left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2 - \left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 \right| \le \widetilde{\delta}_{2s+1} \|\boldsymbol{r}\|_2, \\
&\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 - \widetilde{\delta}_{2s+1}\|\boldsymbol{r}\|_2 \le \left\|\frac{\widetilde{\boldsymbol{r}}}{m_1}\right\|_2 \le \left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 + \widetilde{\delta}_{2s+1}\|\boldsymbol{r}\|_2. \qquad (4.40)
\end{aligned}$$

Combining (4.39) and (4.40), we reach the conclusion in (4.36). □

**Lemma 8.** *Let $\mathcal{H}_{2s}$ be the set of all functions $h$ whose coefficient vectors are $2s$-sparse and let $\boldsymbol{r}_h \in \mathbb{C}^N$ denote the coefficient vector for each $h \in \mathcal{H}_{2s}$. Let $j$ be any integer such that*

*$j \in [D] \setminus \{0\}$, and $\widetilde{\delta}$ and $\delta'$ be chosen from $(0,1)$. Given $\alpha > 1$, assume that the restricted $B_j$ and $C_j$ satisfy RIP with $\widetilde{\delta}_{2s+1} \le \widetilde{\delta} \le \frac{1}{\check{c}\sqrt{s}}$ for some $\check{c} > \sqrt{2}$ and $\delta'_{2s} \le \delta' \in (0,1)$, respectively, with $\sqrt{\frac{1+\delta'}{1-\delta'}} < \frac{\check{c}}{\alpha} - 1$. Then, denoting $\boldsymbol{r} := \boldsymbol{r}_h$ for simplicity, one has the set $\mathcal{P}_j \supset \left\{ \widetilde{\boldsymbol{n}} \in [M]^{j+1} \mid \|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2 \ge \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$ of cardinality $2s$ resulting from Algorithm 7 if $\mathcal{P}_{j-1} \supset \left\{ \hat{\boldsymbol{n}} \in [M]^j \mid \|\boldsymbol{r}_{j-1;(\hat{\boldsymbol{n}},\cdots)}\|_2^2 \ge \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$.*

*Proof.* By assumption that $\mathcal{P}_{j-1} \subset \times_{i\in[j]}\mathcal{N}_i$ contains all prefixes in $\{\hat{\boldsymbol{n}} \in [M]^j \mid \|\boldsymbol{r}_{j-1;(\hat{\boldsymbol{n}},\cdots)}\|_2^2 \ge \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}\}$, $\mathcal{P}_{j-1} \times \mathcal{N}_j$ contains all possible prefixes $\widetilde{\boldsymbol{n}} \in [M]^{j+1}$ with $\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2 \ge \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}$ by the definition of $\mathcal{P}_j$ and $\mathcal{N}_j$ for all $j \in [D]$. If $\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 = 0$, i.e., there is no $\boldsymbol{n}_2$ such that $(\widetilde{\boldsymbol{n}}, \boldsymbol{n}_2) \in \operatorname{supp}(\boldsymbol{r})$, then from Lemma 7,

$$0 \le E_{j;(\widetilde{\boldsymbol{n}},\cdots)} \le (1+\delta')\left(\widetilde{\delta}\|\boldsymbol{r}\|_2\right)^2 \le (1+\delta')\left(\frac{\|\boldsymbol{r}\|_2}{\check{c}\sqrt{s}}\right)^2.$$

On the other hand, if $\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2^2 \ge \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}$, i.e., there is $\boldsymbol{n}_2$ such that $(\widetilde{\boldsymbol{n}}, \boldsymbol{n}_2) \in \operatorname{supp}(\boldsymbol{r})$, then

$$\begin{aligned}(1-\delta')\left(\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 - \frac{\|\boldsymbol{r}\|_2}{\check{c}\sqrt{s}}\right)^2 &\le (1-\delta')\left(\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 - \widetilde{\delta}\|\boldsymbol{r}\|_2\right)^2 \\ &\le E_{j;(\widetilde{\boldsymbol{n}},\cdots)} \le (1+\delta')\left(\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 + \widetilde{\delta}\|\boldsymbol{r}\|_2\right)^2.\end{aligned}$$

In order to distinguish nonzero $\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2 \ge \frac{\|\boldsymbol{r}\|_2}{\alpha\sqrt{s}}$ from zero $\left\|\boldsymbol{r}_{j;(\widetilde{\boldsymbol{n}},\cdots)}\right\|_2$, we should have

$$(1+\delta')\left(\frac{\|\boldsymbol{r}\|_2}{\check{c}\sqrt{s}}\right)^2 < (1-\delta')\left(\frac{\|\boldsymbol{r}\|_2}{\alpha\sqrt{s}} - \frac{\|\boldsymbol{r}\|_2}{\check{c}\sqrt{s}}\right)^2$$

which is implied by

$$\sqrt{\frac{1+\delta'}{1-\delta'}} < \frac{\check{c}}{\alpha} - 1,$$

as in the assumption. Since estimates of zero energy and nonzero energy greater than $\frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}$ are separated, choosing $2s$ prefixes with largest estimates $E_{\widetilde{\boldsymbol{n}}}$ guarantees that it contains all prefixes with energy greater than $\frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}$. $\square$

**Theorem 10.** *Let $\mathcal{H}_{2s}$ be the set of all functions $h$ whose coefficient vectors are $2s$-sparse and let $\boldsymbol{r}_h \in \mathbb{C}^N$ denote the coefficient vector for each $h \in \mathcal{H}_{2s}$. We assume that we have $\mathcal{N}_j$ for all $j \in [D]$. Let $\alpha > 1$, $\widetilde{\delta} \leq \frac{1}{\check{c}\sqrt{s}}$ for some $\check{c} > \sqrt{2}$ and $\delta' \in (0,1)$, satisfying $\sqrt{\frac{1-\delta'}{1+\delta'}} < \frac{\check{c}}{\alpha} - 1$, and $p \in (0,1)$. If*

$$m_1 \geq \bar{\alpha} K^2 \left(\widetilde{\delta}\right)^{-2} s \max\left\{\log^2(4s) \cdot d \cdot \log\left(\frac{\mathrm{e} M D^{1+\frac{1}{d\log^3(4s)}}}{d}\right)\log(9m_1), \log(2D/p)\right\} \text{ and}$$

$$m_2 \geq \bar{\beta} K^2 \left(\delta'\right)^{-2} s \max\left\{\log^2(4s) \cdot d \cdot \log\left(\frac{\mathrm{e} M D^{1+\frac{1}{d\log^3(4s)}}}{d}\right)\log(9m_2), \log(2D/p)\right\},$$

*for absolute constants $\bar{\alpha}$ and $\bar{\beta}$, then denoting $\boldsymbol{r} := \boldsymbol{r}_h$ for simplicity, Algorithm 7 finds $\mathcal{P} \supset \left\{\boldsymbol{n} \in [M]^D \mid |r_{\boldsymbol{n}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}\right\}$ of $|\mathcal{P}| = 2s$ with probability at least $1 - p$.*

*Proof.* Given $m_1$ and $m_2$, by Theorem 3, the probability of either $B_j$ or $C_j$ not satisfying $\widetilde{\delta}_{2s+1} < \widetilde{\delta}$ or $\delta'_{2s} < \delta'$ respectively is at most $\frac{p}{2D}$ for each $j \in [D] \setminus \{0\}$, and thus the union bound over all $j$ yields the failure probability at most $\frac{p(D-1)}{D} < p$. That is, Theorem 3 ensures that $B_j$ and $C_j$ have RIP uniformly for all $j \in [D] \setminus \{0\}$ with probability at least $1 - p$. Repeatedly applying Lemma 8 yields the final $\mathcal{P} (= \mathcal{P}_{D-1}) \supset \left\{\boldsymbol{n} \in [M]^D \;\middle|\; |r_{\boldsymbol{n}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}\right\}$ of cardinality $2s$ by combining the fact that $\mathcal{P}_0 (= \mathcal{N}_0) \supset \left\{\widetilde{\boldsymbol{n}} \in [M] \;\middle|\; \|\boldsymbol{r}_{(\widetilde{\boldsymbol{n}},\cdots)}\|_2^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}\right\}$. $\square$

### 4.3.3 Support Identification

In this section, it remains to combine the results of entry identification and pairing processes in order to give the complete support identification algorithm and to prove Lemma 2 which is the main ingredient of Theorem 7 in Section 4.2.3. The support identification starts with the entry identification providing $\mathcal{N}_j$, $j \in [D]$ as outputs, and in turn, the pairing takes $\mathcal{N}_j$, $j \in [D]$ as inputs and outputs $\mathcal{P}$ of cardinality $2s$ containing $\left\{ \boldsymbol{n} \in [M]^D \mid |r_{\boldsymbol{n}}|^2 \geq \frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s} \right\}$. Accordingly, we can get the following result.

**Lemma 9.** *The set $\mathcal{P}$ contains at most $2s$ index vectors satisfying*

$$\|\boldsymbol{r}_{\mathcal{P}^c}\|_2 \leq \sqrt{2s\frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}} = \frac{\sqrt{2}\|\boldsymbol{r}\|_2}{\alpha},$$

*where $\boldsymbol{r}_{\mathcal{P}^c}$ is the vector of $\boldsymbol{r}$ restricted to the complement of $\mathcal{P}$.*

*Proof.* Note that $\boldsymbol{r}$ is $2s$-sparse and by Theorem 10 the squared magnitude of each $r_{\boldsymbol{n}}$ at $\mathcal{P}^c$ is less than $\frac{\|\boldsymbol{r}\|_2^2}{\alpha^2 s}$ so that we obtain the desired result. $\square$

Finally, we are ready to prove Lemma 2 in order to complete the analysis of support identification.

*Proof of Lemma 2.* By choosing $\alpha = 7$ and $\mathcal{P} = \Omega$ in Lemma 9, we obtain the desired upper bound of $\|(\boldsymbol{r}_h)_{\Omega^c}\|_2$ with probability at least $1 - 2p$ given the grids $\mathcal{G}^I$ and $\mathcal{G}^P$. The union bound of the failure probabilities $p$ of Lemma 6 and Theorem 10 gives the desired probability. It remains to demonstrate the sampling complexity combining $\left|\mathcal{G}^I\right|$ and $\left|\mathcal{G}^P\right|$, and the runtime complexity combining Algorithms 6 and 7. The first term $\left|\mathcal{G}^I\right|$ is $m\mathcal{L}'D$ where $m$ comes from Lemma 6, $\mathcal{L}'$ is defined as in Section 4.2.3 , and $D$, the number of changes in $j \in [D]$, therein. We emphasize that $m$ samples are utilized repeatedly in order to implicitly construct $L$ proxy functions combined with different Gaussian weights so that $L$ does not affect the sampling complexity but affects the runtime complexity. The second term $\left|\mathcal{G}^P\right|$ is $m_1 m_2(D-2) + m_1$ where $m_1$ and $m_2$ are from Theorem 10, and $D-2$ is the number of changes in $j \in [D-1] \setminus \{0\}$. We remind readers that $C_{D-1} = 1$ implied by $\mathcal{Z}_{D-1} = \emptyset$ so that $m_1$ samples are utilized instead of $m_1 m_2$ when $j = D-1$. Now, we consider the runtime complexity. The first term in runtime complexity is $\mathcal{O}(mL\mathcal{L}D)$ from Algorithm 6 where $mL$ computations are taken to implicitly construct the $L$ proxy functions, $\mathcal{O}(\mathcal{L})$ is defined as in Section 4.2.3 , and $D$ comes from the for loop from Algorithm 6. The second term in runtime complexity is $4s^2\left(m_1 m_2(D-2) + m_1\right) + 4s^2 m_1 \sum_{j=1}^{D-1}(j+1)$ from Algorithm 7 since $4s^2$ energy estimates are calculated using the $m_1 m_2$ samples for each $j \in [D-1] \setminus \{0\}$ and $m_1$ samples for $j = D-1$, and the evaluations of $T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_{j,\ell})$ are calculated for all $\widetilde{\boldsymbol{n}} \in \mathcal{P}_j$, $\forall j \in [D] \setminus \{0\}$. Here, it is assumed that it takes $\mathcal{O}(1)$ runtime to evaluate each $i^{\text{th}}$ component $T_{i;\widetilde{n}_i}((w_{j,\ell})_i)$ of $T_{j;\widetilde{\boldsymbol{n}}}(\boldsymbol{w}_{j,\ell})$. $\square$

## 4.4 Empirical Evaluation

In this section Algorithm 5 is evaluated numerically and compared to CoSaMP [22], its superlinear-time progenitor. Both Algorithm 5 and CoSaMP were implemented in MATLAB for this purpose.

### 4.4.1 Experimental Setup

We consider two kinds of tensor product basis functions below: Fourier and Chebyshev. In both cases each parameter, $M$, $D$, and $s$, is changed while the others remain fixed so that we can see how each parameter affects the runtime, sampling number, memory usage, and error of both Algorithm 5 and CoSaMP. For all experiments below $d = D$ so that $\mathcal{I} = [M]^D$. Every data point in every plot below was created using 100 different randomly generated trial signals, $f$, of the form

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{n} \in \mathcal{S}} c_{\boldsymbol{n}} T_{\boldsymbol{n}}(\boldsymbol{x}), \tag{4.41}$$

where each function's support set, $\mathcal{S}$, contained $s$ index vectors $\boldsymbol{n} \in [M]^D$ each of which was independently chosen uniformly at random from $[M]^D$, and where each function's coefficients $c_{\boldsymbol{n}}$ were each independently chosen uniformly at random from the unit circle in the complex plane (i.e., each $c_{\boldsymbol{n}} = \mathbb{e}^{i\theta}$ where $\theta$ is chosen uniformly at random from $[0, 2\pi]$). In the Fourier setting the basis functions $T_{\boldsymbol{n}}(\boldsymbol{x})$ in (4.41) were chosen as per (4.42), and in the Chebyshev setting as per (4.43).

Below a *trial* will always refer to the execution of Algorithm 5 and/or CoSaMP on a particular randomly generated trial function $f$ in (4.41). A *failed trail* will refer to any trial where either CoSaMP or Algorithm 5 failed to recover the correct support set $\mathcal{S}$ for $f$. Herein the parameters of both Algorithm 5 and CoSaMP were tuned to keep the number of failed trials down to less than 10 out of the total 100 used to create every datapoint in every plot. Finally, in all of our plots Algorithm 5 is graphed with red, and CoSaMP with blue.

### 4.4.2 Experiments with the Fourier Basis for $\mathcal{D} = [0, 1]^D$

In this section we consider the Fourier tensor product basis

$$T_{\boldsymbol{n}}(\boldsymbol{x}) := \prod_{j=0}^{D-1} \mathbb{e}^{2\pi \mathbb{i} n_j x_j} \tag{4.42}$$

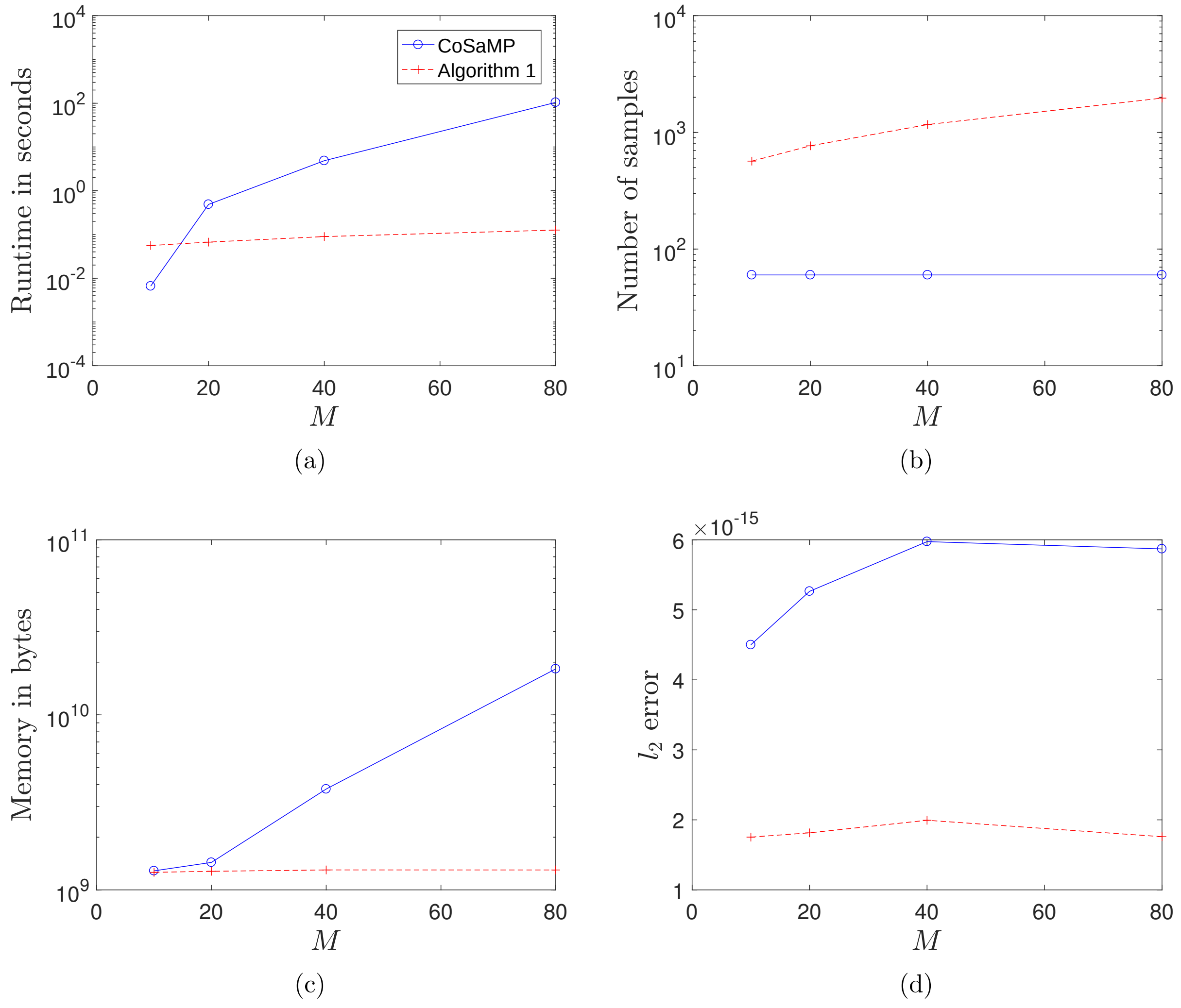
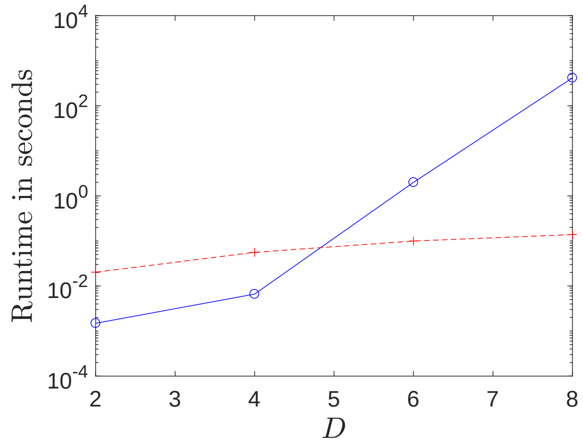
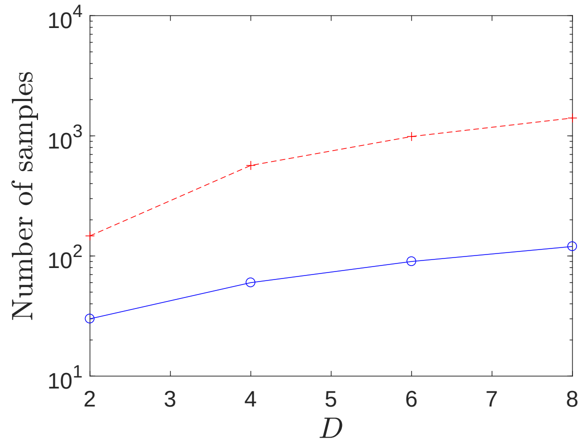


Figure 4.1: Fourier basis, $M \in \{10, 20, 40, 80\}$, $D = 4$, $s = 5$

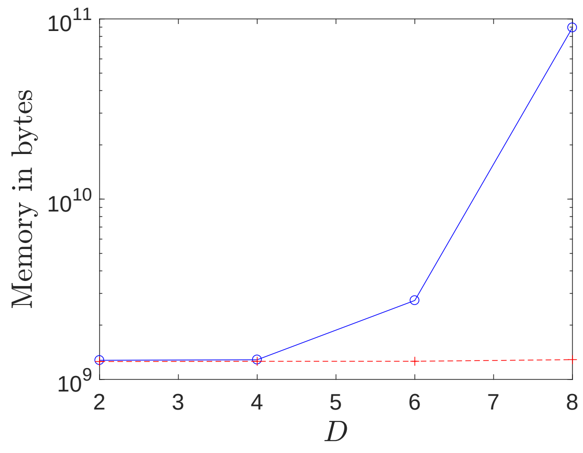
whose orthogonality measure is the Lebesgue measure on $\mathcal{D} = [0, 1]^D$. In figures 4.1, 4.2 and 4.3, results are shown for approximating Fourier-sparse trial functions (4.41) using noiseless samples $\boldsymbol{y}$. In figure 4.1, the parameter $M$ changes over the set $\{10, 20, 40, 80\}$ while $D = 4$ and $s = 5$ are held constant. In figure 4.1a, the average runtime (in seconds) is shown as $M$ changes. The average here is calculated over all 100 trials at each data point excluding any failed trials. As we can see, the runtime of Algorithm 5 grows very slowly as $M$ grows, whereas the runtime grows fairly quickly for CoSaMP since its measurement matrix's size



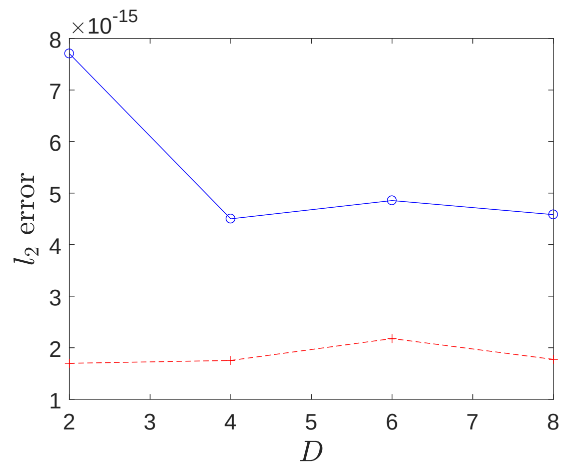
(a)



(b)



(c)



(d)

Figure 4.2: Fourier basis, $M = 10,\ D = \{2, 4, 6, 8\},\ s = 5$

increases significantly as $M$ grows. Figure 4.1b shows the number of samples used by both CoSaMP and Algorithm 5. We can see that Algorithm 5 requires more samples due mainly to its support identification's pairing step. On the other hand, we can see in figure 4.1c that the memory usage of CoSaMP grows very rapidly compared to the slow growth of Algorithm 5's memory usage. This exemplifies the tradeoff between Algorithm 5 and CoSaMP – Algorithm 5 uses more samples than CoSaMP in order to reduce its runtime complexity and memory usage for large $D$ and $M$. Finally, figure 4.1d demonstrates that both methods

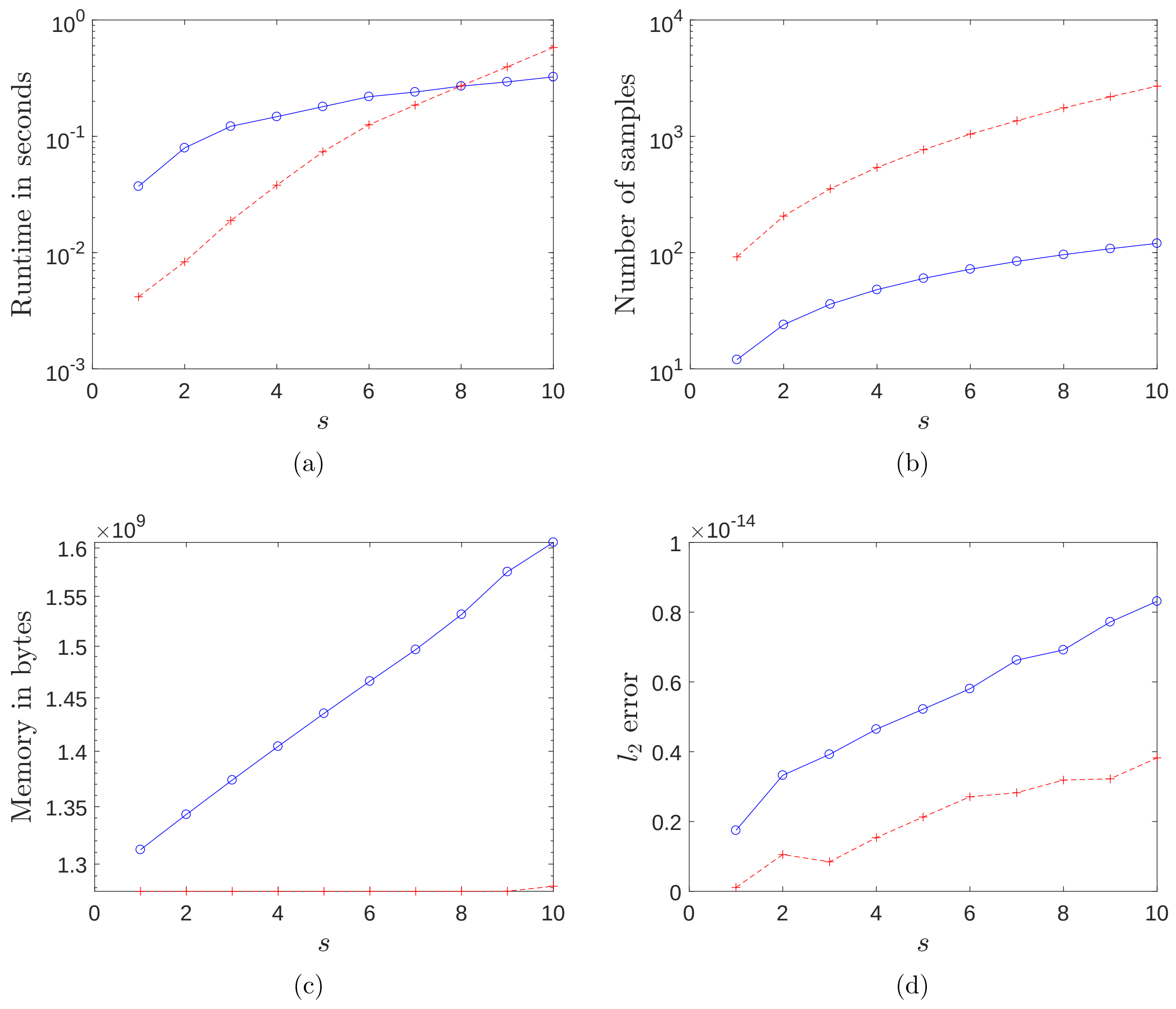


Figure 4.3: Fourier basis, $M = 20,\ D = 4,\ s = \{1, 2, 3, \cdots, 10\}$

produce outputs whose average errors (over the trials where they don't fail) are on the order of $10^{-15}$.

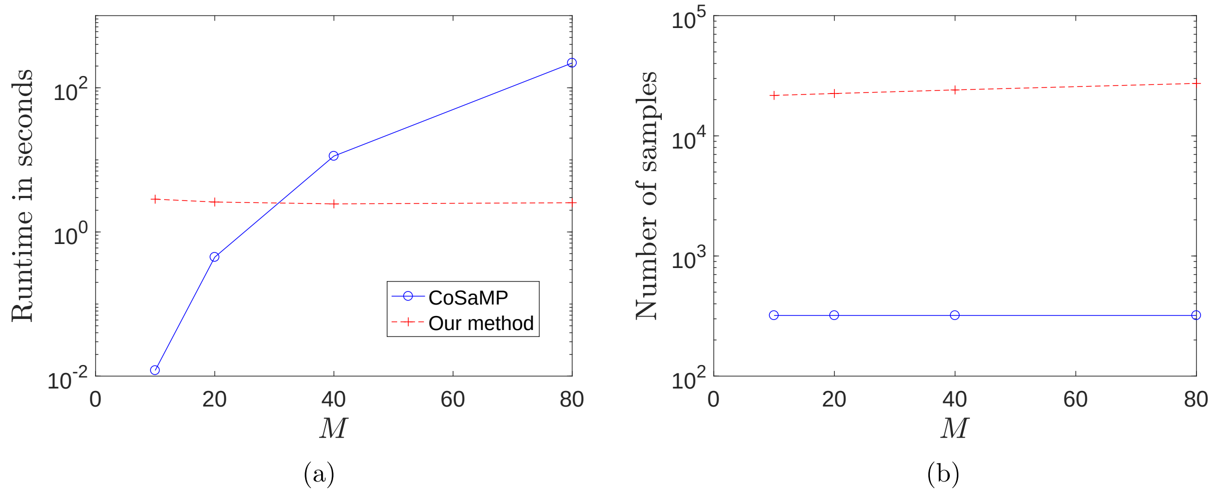
In figure 4.2, the number of dimensions, $D$, changes while both $M = 10$ and $s = 5$ are held fixed. Here, we can clearly see the advantage of Algorithm 5 for functions of many variables. The runtime and memory usage of CoSaMP blow up quickly as $D$ increases due to the gigantic matrix-vector multiplies it requires to identify support. Algorithm 5, on the other hand, shows much slower growth in runtime and memory usage. When $D = 10$, for example, CoSaMP requires terabytes of memory whereas Algorithm 5 requires only a few gigabytes. In figure 4.3, $s$ varies in $\{1, 2, 3, \cdots, 10\}$ while $M = 20$ and $D = 4$ are fixed. Since Algorithm 5 has $\tilde{\mathcal{O}}(s^5)$ scaling in runtime due to its pairing step, it suffers as sparsity increases more quickly than CoSaMP does. Note that the crossover point is around $s = 8$, so that Algorithm 5 appears to be slower than CoSaMP for all $s > 8$ when $M = 20$ and $D = 4$. Though "only polynomial in $s$", it is clear from these experiments that the runtime scaling of Algorithm 7 in $s$ needs to be improved before the methods proposed herein can become truly useful in practice.

### 4.4.3 Experiments with the Chebyshev Basis for $\mathcal{D} = [-1, 1]^D$

In this section we consider the Chebyshev tensor product basis

$$T_{\boldsymbol{n}}(\boldsymbol{x}) := 2^{\frac{1}{2}\|\boldsymbol{n}\|_0} \prod_{j=0}^{D-1} \cos\left(n_j \arccos(x_j)\right) \tag{4.43}$$

whose orthogonality measure is $d\boldsymbol{\nu} = \otimes_{j\in[D]} \frac{dx_j}{\pi\sqrt{1-x_j^2}}$ on $\mathcal{D} = [-1, 1]^D$. Runtime, memory, sampling complexity, and error graphs are provided in figures 4.4, 4.5 and 4.6 as $M$, $D$, and $s$ vary, respectively. Since this Chebyshev product basis has a BOS constant of $K = 2^{D/2}$, both CoSaMP and Algorithm 5 suffer from a mild exponential grown in sampling, runtime, and memory complexity as $D$ increases (recall that $d = D$ for these experiments). This leads to markedly different overall performance for the Chebyshev basis than what is observed for the Fourier basis where $K = 1$. A reduction in performance from the Fourier case for both methods is clearly visible, e.g., in figure 4.5. Nonetheless, Algorithm 5 demonstrates the expected reduced runtime and sampling complexity dependence on $M$ and $D$ over CoSaMP in figures 4.4 and 4.5, as well as a striking reduction in its required memory usage over



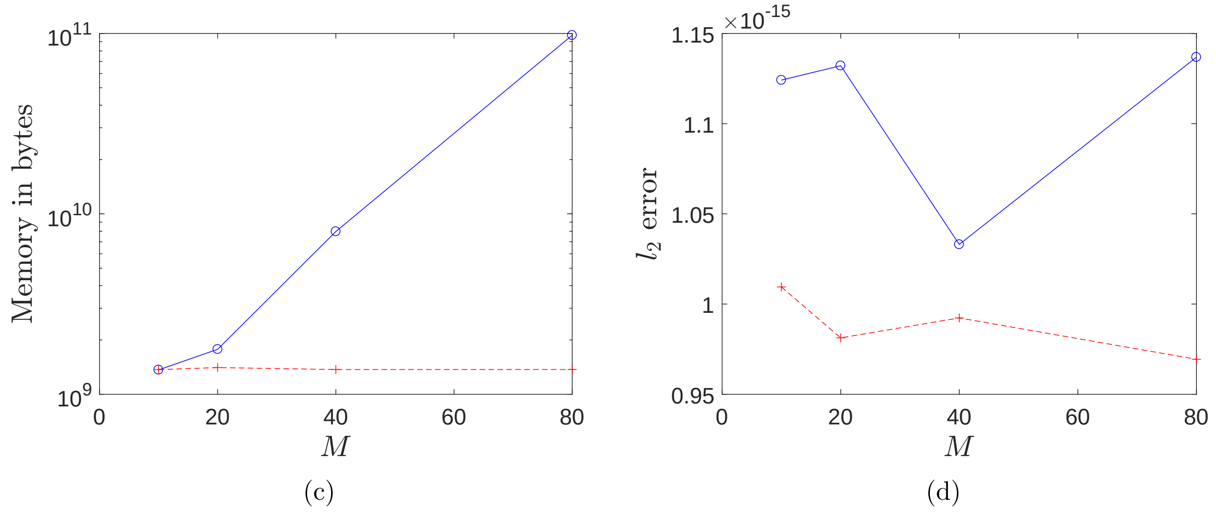


Figure 4.4: Chebyshev basis, $M \in \{10, 20, 40, 80\}$, $D = 4$, $s = 5$

CoSaMP even in figure 4.6 when its runtime complexity is worse. Unfortunately, the $\tilde{\mathcal{O}}(s^5)$ runtime dependance of Algorithm 7 on sparsity is again clear in figure 4.6a leading to a crossover point of Algorithm 5 with CoSaMP at only $s = 3$ when $M = 10$ and $D = 6$. This again clearly marks the pairing process of Algorithm 7 as being in need of improvement.

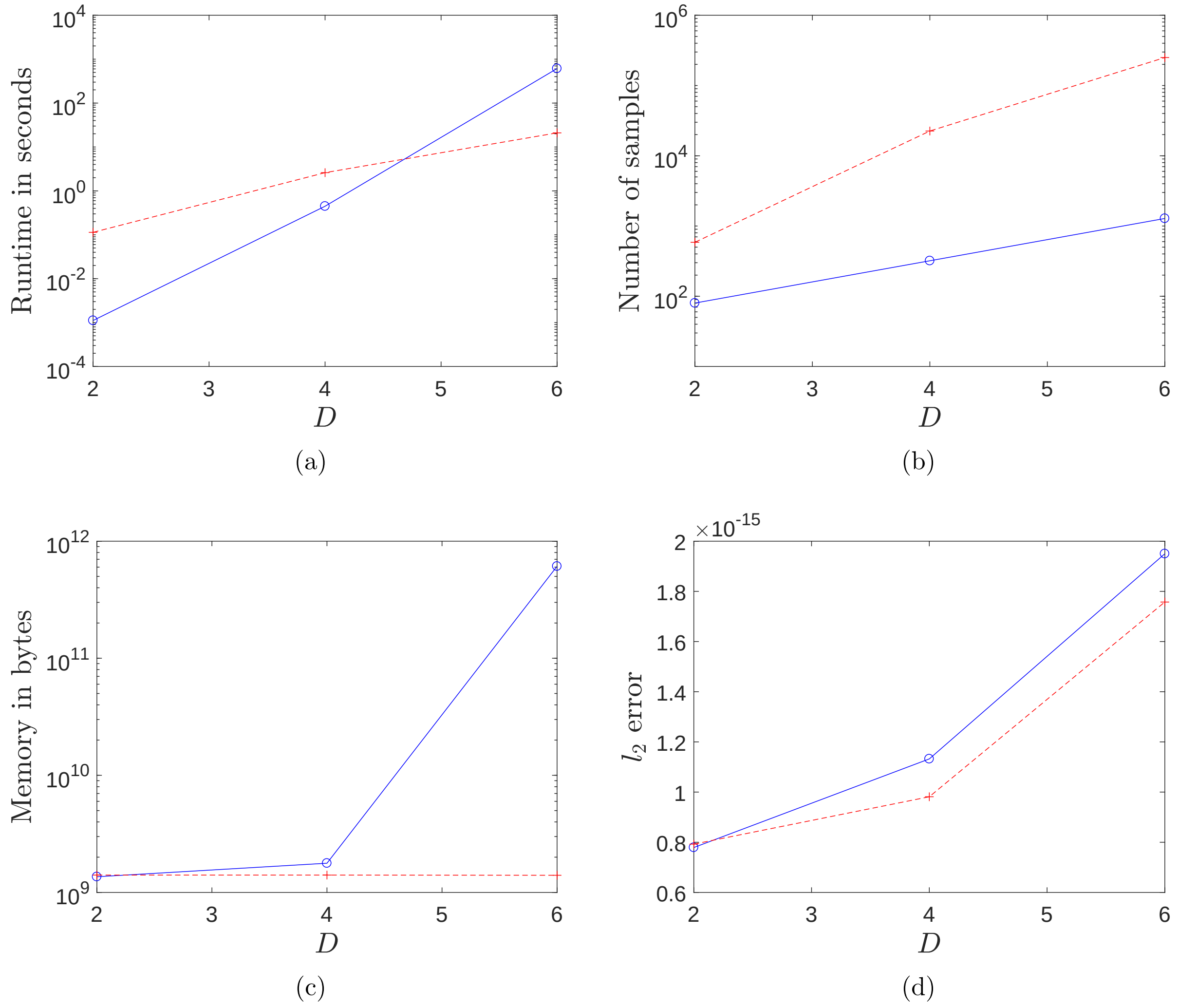


Figure 4.5: Chebyshev basis, $M = 20,\ D = \{2, 4, 6\},\ s = 5$

### 4.4.4 Experiments for Larger Ranges of Sparsity $s$ and Dimension $D$

Figures 4.7 and 4.8 explore the performance of Algorithm 5 on Fourier sparse functions for larger ranges of $D$ and $s$, respectively. In figure 4.7, a function of $D = 75$ variables can be recovered in just a few seconds when it is sufficiently sparse in the Fourier basis. It is worth pointing out here that when $D = 75$ the BOS in question contains $20^{75} \sim 10^{97}$ basis functions, significantly more than the approximately $10^{82}$ atoms estimated to be in

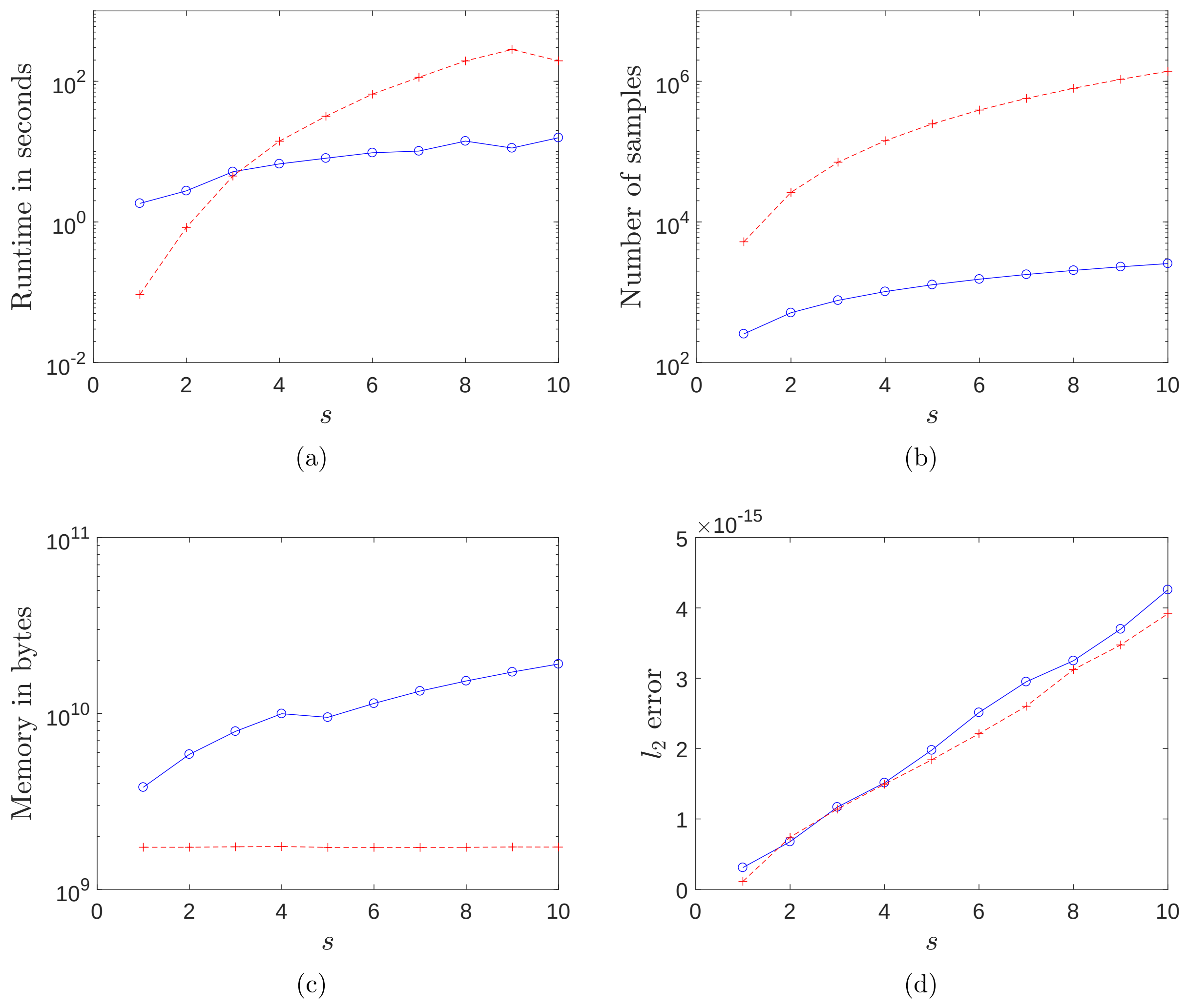


Figure 4.6: Chebyshev basis, $M = 10,\ D = 6,\ s = \{1, 2, 3, \cdots, 10\}$

the observable universe. We would like to emphasize that Algorithm 5 is solving problems in this setting that are simply too large to be solved efficiently, if at all, using standard superlinear-time compressive sensing approaches due to their memory requirements when dealing with such extremely large bases. Figure 4.8 also shows that functions with larger Fourier sparsities, $s$, than previously considered (up to $s = 160$) can be be recovered in about an hour or less from a BOS of size $40^5 = 102,400,000$.

In figures 4.9 and 4.10 we consider the functions which are sparse in the Chebyshev prod-

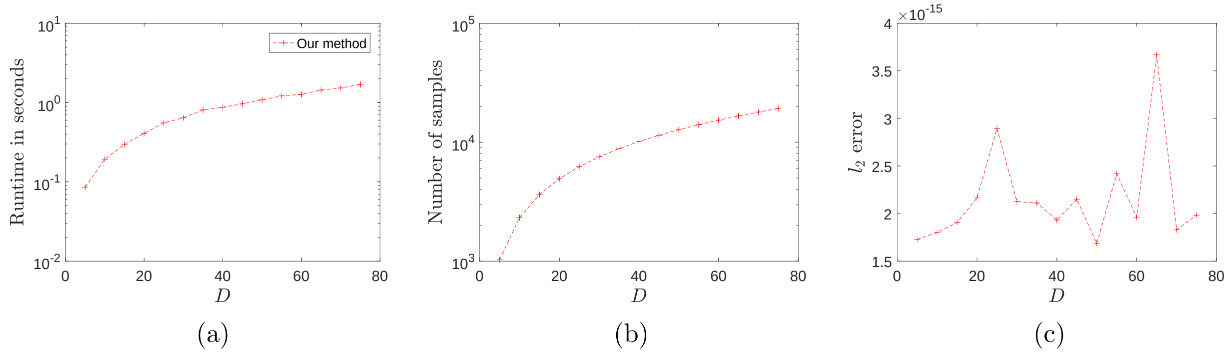


Figure 4.7: Fourier basis, $M = 20$, $D \in \{5, 10, 15, 20, \cdots, 75\}$, $s = 5$

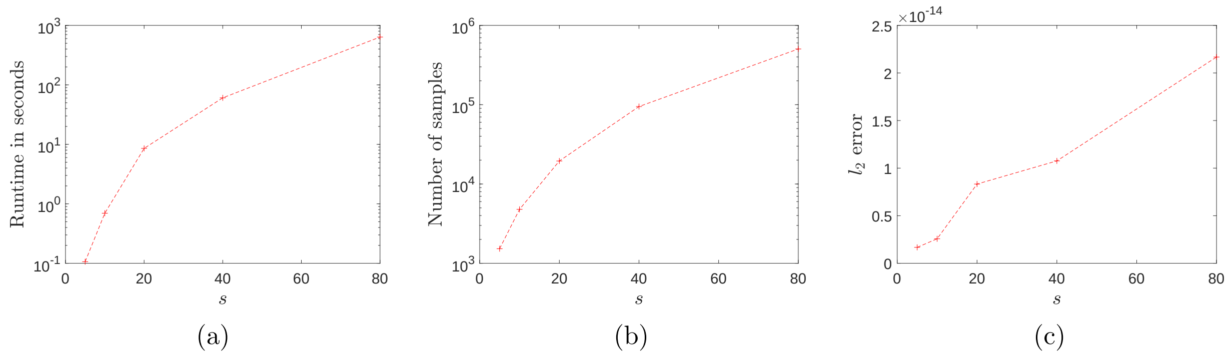


Figure 4.8: Fourier basis, $M = 40$, $D = 5$, $s \in \{5, 10, 20, 40, 80\}$

uct basis. Again, due to the larger BOS constant of the Chebyshev basis, the $D$ and $s$ ranges that our method can deal efficiently are smaller than in the Fourier case. When $D$ is 12 or $s$ is 20 in figures 4.9 and 4.10, respectively, for example, it takes a few hours for Algorithm 5 to finish running. We again remind the readers that standard superlinear-time compressive sensing methods cannot solve with such high dimensional problems at all, however, on anything less than a world class supercomputer due to their memory requirements. In the figure 4.9 experiments the Chebyshev BOS contains $20^{12} \sim 10^{15}$ basis functions when $D = 12$. In the figure 4.10 experiments the BOS contains just over 100 million basis functions.

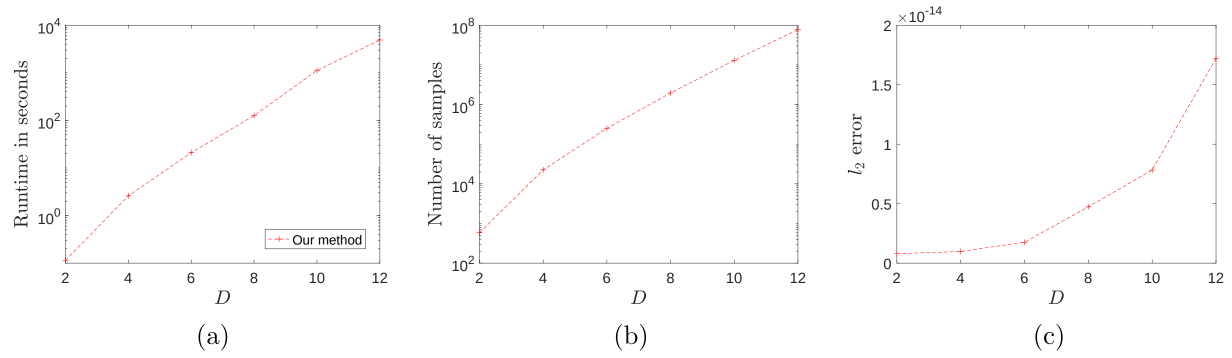


Figure 4.9: Chebyshev basis, $M = 20,\ D \in \{2, 4, 6, \cdots, 12\},\ s = 5$

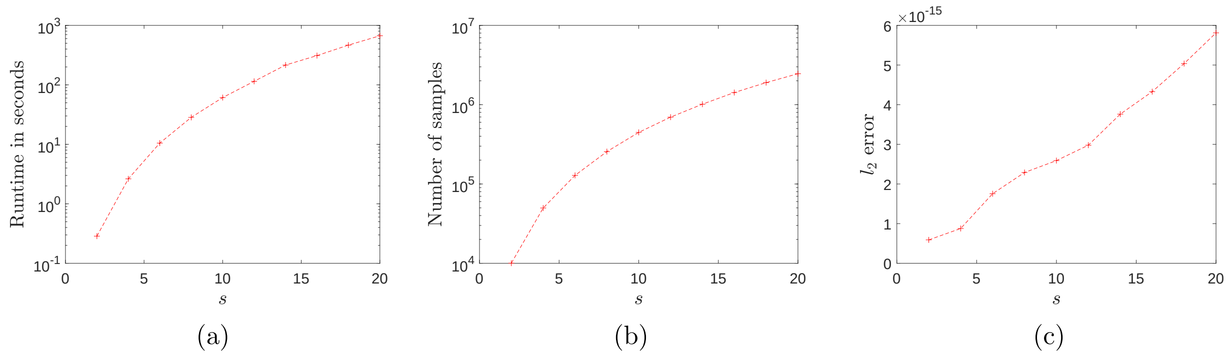


Figure 4.10: Chebyshev basis, $M = 40,\ D = 5,\ s \in \{2, 4, 6, \cdots, 20\}$

### 4.4.5 Recovery of Functions from Noisy Measurements

In figures 4.11 and 4.12 we further consider exactly sparse trial functions (4.41) whose function evaluations are contaminated with Gaussian noise. That is, we provide Algorithm 5 with noisy samples

$$\boldsymbol{y}' \ = \ \boldsymbol{y} + \boldsymbol{g}' \ = \ \boldsymbol{y} + \sigma \frac{\|\boldsymbol{y}\|^2}{\|\boldsymbol{g}\|_2} \boldsymbol{g}$$

where $\boldsymbol{y}$ contains noiseless samples from each $f$ as per (4.4), $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, I)$, and $\sigma \in \mathbb{R}^+$ is used to control the Signal to Noise Ratio (SNR) defined herein by

$$\mathrm{SNR}_{\mathrm{db}} := 10 \log_{10} \left( \frac{\|\boldsymbol{y}\|_2^2}{\|\boldsymbol{g}'\|_2^2} \right) \ = \ -10 \log_{10} \left( \sigma^2 \right).$$

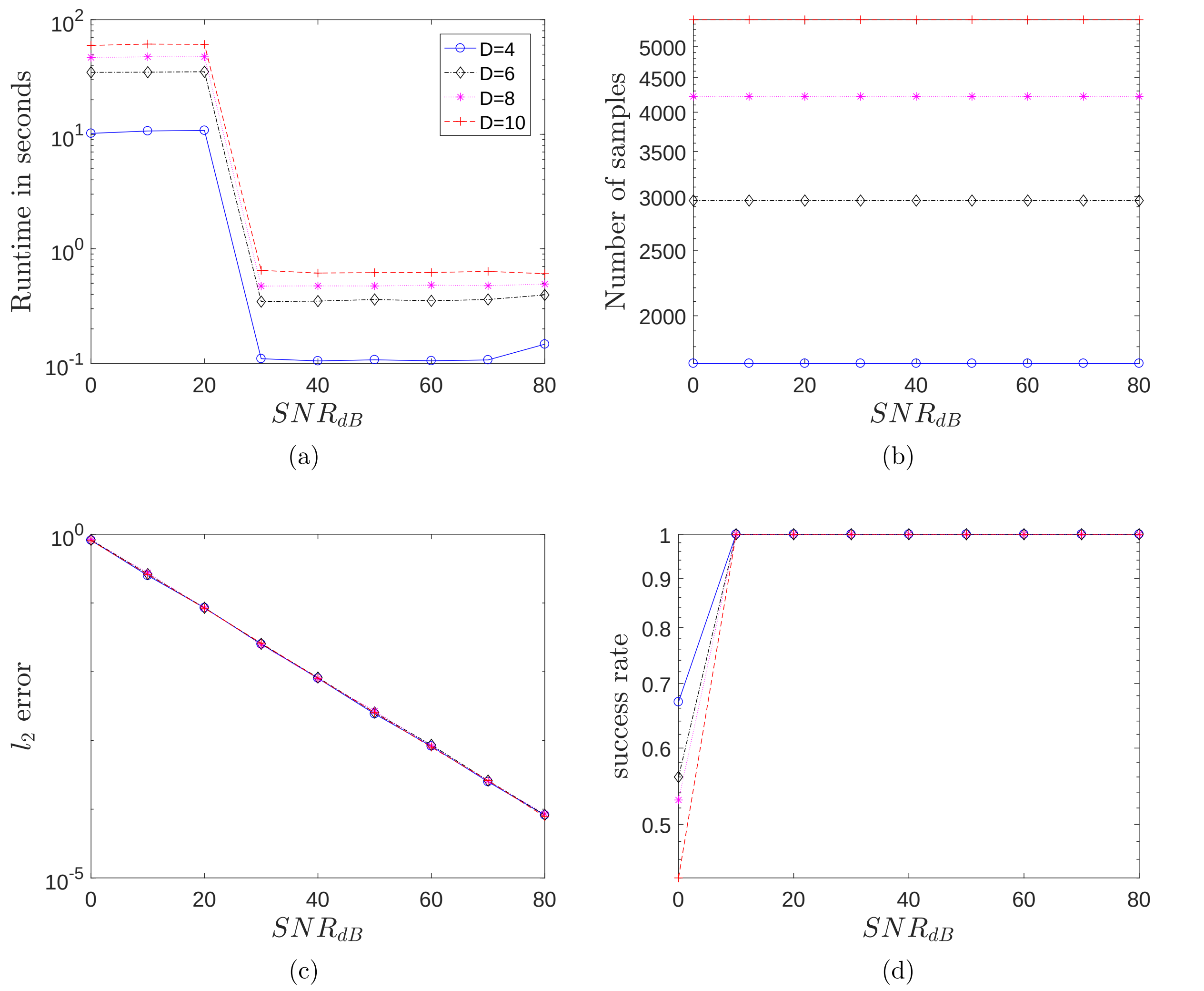


Figure 4.11: Algorithm 5, Fourier basis, $M = 10$, $D \in \{4, 6, 8, 10\}$, $s = 5$, $\text{SNR}_{dB} \in \{0, 10, 20, \cdots, 80\}$

Figures 4.11 and 4.12 show the performance of Algorithm 5 for the Fourier and Chebyshev product bases, respectively, as SNR varies. Figure 4.11a shows the average runtime for each $D \in \{2, 4, 6, 8\}$ as $\text{SNR}_{dB}$ changes. When $\text{SNR}_{dB}$ is close to 0 (which means that the $\ell_2$-norm of noise vector is the same as the $\ell_2$-norm of sample vector), the runtime gets larger due to Algorithm 5 using a larger number of overall iterations. The runtime also increases mildly as $D$ increases in line with our previous observations. The sampling number in Figure 4.11b is set to be three times larger than the sampling number used in the noiseless cases. Figure

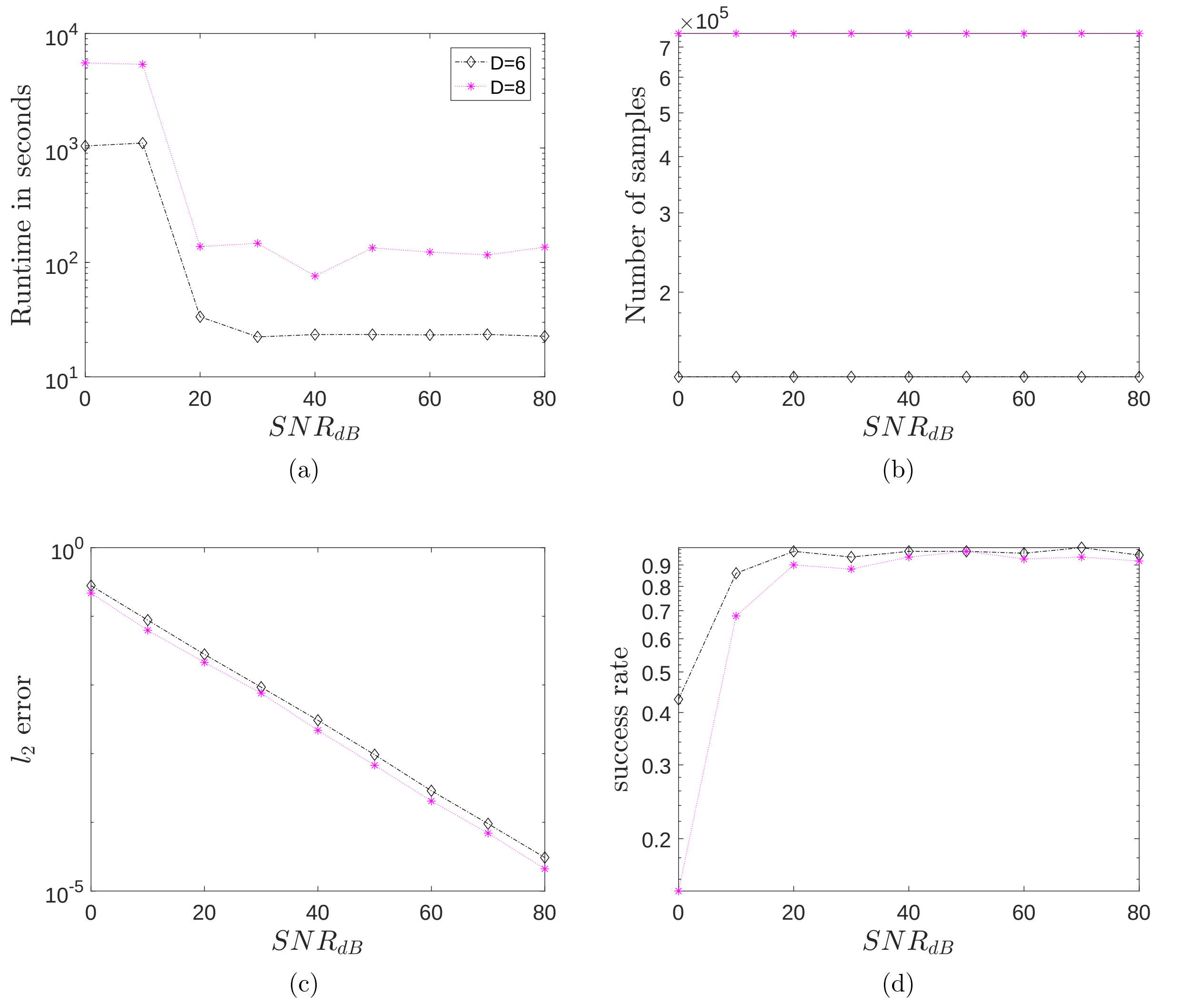


Figure 4.12: Algorithm 5, Chebyshev basis, $M = 10$, $D \in \{6, 8\}$, $s = 5$, $\text{SNR}_{dB} \in \{0, 10, 20, \cdots, 80\}$

4.12 shows the results of Algorithm 5 applied to functions which are sparse in the Chebyshev product basis. Similar to the Fourier case, the runtime grows as the noise level gets worse in Figure 4.12a. Also, larger $D$ results in the larger runtime as previously discussed. In Figure 4.12b, the sampling number is also set by tripling the sampling number used in noiseless cases.

As above, in both figures 4.11 and 4.12 the average $\ell_2$-error is computed by only considering the successful trials where every element of $f$'s support, $\mathcal{S}$, is found. Here, however, the percentage of successful trials falls below 90% for lower SNR values. The success rates (i.e., the percentage of successful trials at each data point) are therefore plotted in figures 4.11d and 4.12d. Both figures show that a smaller $\text{SNR}_{dB}$ results in a smaller success rate, as one might expect. As $\text{SNR}_{dB}$ increases, however, the $\ell_2$-error decreases linearly for the successful trials.

### 4.4.6 Some Additional Implementational Details

In the line 13 of Algorithm 5 solving the least square problem can be accelerated by the iterative algorithms such as the Richardson method or the conjugate gradient method when the size of the matrix $\Phi_T$ is large [22]. For our range of relatively low sparsities, however, there was not much difference in the runtime between using such iterative least square solving algorithms and simply multiplying $\boldsymbol{y}^{\mathbf{E}}$ by the Moore-Penrose inverse, $\Phi_T^{\dagger} := (\Phi_T^* \Phi_T)^{-1} \Phi_T^*$. Thus, we simply form and use the Moore-Penrose inverse for both CoSaMP and Algorithm 5 in our implementations below.

Similarly, in our CoSaMP implementation the conjugate transpose of the measurement matrix, $\Phi$, of size $m \times M^D$ is simply directly multiplied by the updated sample vector $\boldsymbol{v}$ in each iteration in order to obtain the signal proxy used for CoSaMP's support identification procedure (recall that $d = D$ in all experiments below so that $\mathcal{I} = [M]^D$). It is important to note that this matrix-vector multiplication can generally be done more efficiently if, e.g., one instead uses nonuniform FFT techniques [56] to evaluate $\Phi^* \boldsymbol{y}$ for the types of high-dimensional Fourier and Chebyshev basis functions considered below. However, such techniques are again not actually faster than a naive direct matrix multiply for the ranges of relatively low sparsities we consider in the experiments herein.[10] Furthermore, such nonuni-

[10]CoSaMP always uses only $m = \mathcal{O}(s \cdot D \log M)$ samples in the experiments herein which means that its

form FFT techniques will still exhibit exponential runtime and memory dependence on $D$ in the high-dimensional setting even for larger sparsity levels. Thus, nonuniform FFTs were not utilized in our MATLAB implementation of CoSaMP.

measurement matrix's conjugate transpose, $\Phi^* \in \mathbb{C}^{M^D \times m}$, can be naively multiplied by vectors in only $\mathcal{O}(s \cdot D \log M \cdot M^D)$-time. When $s$ is small this is comparable to the $\mathcal{O}(D \log M \cdot M^D)$ runtime complexity of a (nonuniform) FFT.

# Chapter 5

# Conclusion

In this thesis, we developed several sublinear-time high-dimensional function learning algorithms under the assumption that the function has a sparsity in the tensorized basis of bounded orthonormal functions including the tensorized Fourier basis. When focusing on Fourier basis, nice properties of Fourier basis make it possible to more quickly and efficiently recover the frequency vectors and the corresponding Fourier coefficients using much less samples.

In Chapter 2 we showed how to extend our $1D$ sublinear sparse Fourier algorithm to the general $D$ dimensional case. The methods project $D$ dimensional frequency vectors onto lower dimensions. In this process we encounter several obstacles. Thus we introduced "tilting method" for the worst case problems and the "partial unwrapping method" to reduce the chance of collisions and to increase the frequency bandwidth within the limit of computation. Through those methods we can overcome the obstacles as well as maintain the advantage of the $1D$ algorithm. In [1] the average-case sampling complexity is $\mathcal{O}(s)$ and the runtime complexity is $\mathcal{O}(s\log s)$. Extended this estimation from our $1D$ algorithm, we have $\mathcal{O}(Ds)$ sampling complexity and a runtime complexity of $\mathcal{O}(Ds\log s)$ on average under the assumption that there is no worst case scenario happening.

In Chapter 3 we developed a multiscale high-dimensional sparse Fourier algorithm recovering a few energetic Fourier modes using noisy samples. As the estimation error is controlled by the noise level $\sigma$ and the sample length $p$, larger $p$ reduces the error. Rather than recovering the frequencies in a single step, however, we choose multiscale approach in order to make the sample length $p$ increase moderately by improving the estimate iteratively through correction terms determined by a sequence of shifting sizes $\epsilon_q$. We showed that the finite number of correction terms are enough to make the error smaller than 1/2 so that we can reconstruct each integer frequency entry by rounding. As a result, the algorithm has $\mathcal{O}(sD \log M)$ sampling complexity and $\mathcal{O}(sD \log s \log M)$ runtime complexity on average combining the result from Chapter 2 and [2].

In Chapter 4 we have shown that there exist sublinear-time algorithms that approximate multivariate functions sparse in BOS. The approximations provide uniform error guarantee for any function at certain sparsity level. As well as the uniform error guarantee for all exactly sparse functions, the numerical experiments demonstrate our proposed method can well approximate functions containing certain level of noise.

All methods in this thesis assume that we can get the measurement at any sample point. However, this is not always the case in practice. Our future work will be a modification of the algorithms to make it work for given discrete signals with sparsity in Fourier domain using the idea of filtering from [10]. Moreover, one of the future work will be proving rigorous error guarantee of Algorithm 5 for nearly sparse functions. Moreover, combining the work in this section with [32] sublinearizing the dependence on ambient dimension $D$ when the functions only depend on $\widetilde{d}$-variables instead of $D \geq \widetilde{d}$, we expect the further acceleration of approximating functions of many variables.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

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

[2] Andrew Christlieb, David Lawlor, and Yang Wang. A multiscale sub-linear time fourier algorithm for noisy data. *Applied and Computational Harmonic Analysis*, 40(3):553–574, 2016.

[3] Aicke Hinrichs, Erich Novak, Mario Ullrich, and H Woźniakowski. The curse of dimensionality for numerical integration of smooth functions. *Mathematics of Computation*, 83(290):2853–2863, 2014.

[4] Anna C Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 152–161. ACM, 2002.

[5] Anna C Gilbert, S Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse fourier representations. In *Proceedings of SPIE*, volume 5914, page 59141A, 2005.

[6] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012.

[7] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1183–1194. Society for Industrial and Applied Mathematics, 2012.

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

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

[10] Sami Merhi, Ruochuan Zhang, Mark A Iwen, and Andrew Christlieb. A new class of fully discrete sparse fourier transforms: Faster stable implementations with guarantees. *arXiv preprint arXiv:1706.02740*, 2017.

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

[12] Ralph C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013.

[13] Dongbin Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, Princeton, NJ, USA, 2010.

[14] Germund Dahlquist and ke Bjrck. *Numerical Methods in Scientific Computing: Volume 1*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[15] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[16] Russel E Caflisch. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49, 1998.

[17] G. Leobacher and F. Pillichshammer. *Introduction to Quasi-Monte Carlo Integration and Applications*. Compact Textbooks in Mathematics. Springer International Publishing, 2014.

[18] J.-L. Bouchot, H. Rauhut, and C. Schwab. Multi-level Compressed Sensing Petrov-Galerkin discretization of high-dimensional parametric PDEs. *ArXiv e-prints*, January 2017.

[19] Abdellah Chkifa, Nick Dexter, Hoang Tran, and Clayton G Webster. Polynomial approximation via compressed sensing of high-dimensional functions on lower sets. *arXiv preprint arXiv:1602.05823*, 2016.

[20] Holger Rauhut and Christoph Schwab. Compressive sensing petrov-galerkin approximation of high-dimensional parametric operator equations. *Mathematics of Computation*, 86(304):661–700, 2017.

[21] Christoph Schwab and Radu Alexandru Todor. Karhunen–loève approximation of random fields by generalized fast multipole methods. *Journal of Computational Physics*, 217(1):100–122, 2006.

[22] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[23] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[24] MA Iwen, A Gilbert, M Strauss, et al. Empirical evaluation of a sub-linear time sparse dft algorithm. *Communications in Mathematical Sciences*, 5(4):981–998, 2007.

[25] Daniel Potts and Toni Volkmer. Sparse high-dimensional fft based on rank-1 lattice sampling. *Applied and Computational Harmonic Analysis*, 41(3):713–748, 2016.

[26] Michael Kapralov. Sparse fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. *arXiv preprint arXiv:1604.00845*, 2016.

[27] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 261–272, London, UK, UK, 1992. Springer-Verlag.

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

[29] Bosu Choi, Andrew Christlieb, and Yang Wang. Multi-dimensional sublinear sparse fourier algorithm. *arXiv preprint arXiv:1606.07407*, 2016.

[30] Lucia Morotti. Explicit universal sampling sets in finite vector spaces. *Applied and Computational Harmonic Analysis*, 43(2):354–369, 2017.

[31] Daniel Potts and Toni Volkmer. Multivariate sparse fft based on rank-1 chebyshev lattice sampling. In *Sampling Theory and Applications (SampTA), 2017 International Conference on*, pages 504–508. IEEE, 2017.

[32] Ronald DeVore, Guergana Petrova, and Przemyslaw Wojtaszczyk. Approximation of functions of few variables in high dimensions. *Constructive Approximation*, 33(1):125–143, 2011.

[33] Jie Shen and Li-Lian Wang. Sparse spectral approximations of high-dimensional problems based on hyperbolic cross. *SIAM Journal on Numerical Analysis*, 48(3):1087–1109, 2010.

[34] Dinh Dũng and Tino Temlyakov, Vladimir N aiwnd Ullrich. Hyperbolic cross approximation. *arXiv preprint arXiv:1601.03978*, 2016.

[35] B. Choi, A. Christlieb, and Y. Wang. Multi-dimensional Sublinear Sparse Fourier Algorithm. *ArXiv e-prints*, June 2016.

[36] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[37] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing.* Springer, 2013.

[38] AC Gilbert, MJ Strauss, JA Tropp, and R Vershynin. Sublinear approximation of compressible signals. *Proc. SPIE Intell. Integrated Microsystems (IIM)*, page 623, 2006.

[39] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: Fast algorithms for compressed sensing. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 237–246, New York, NY, USA, 2007. ACM.

[40] Anna C Gilbert, Piotr Indyk, Mark Iwen, and Ludwig Schmidt. Recent developments in the sparse fourier transform: a compressed fourier transform for big data. *IEEE Signal Processing Magazine*, 31(5):91–100, 2014.

[41] Sina Bittens, Ruochuan Zhang, and Mark A Iwen. A deterministic sparse fft for functions with structured fourier sparsity. *arXiv preprint arXiv:1705.05256*, 2017.

[42] Mark A Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 20–29. Society for Industrial and Applied Mathematics, 2008.

[43] J Bailey, Mark A Iwen, and Craig V Spencer. On the design of deterministic matrices for fast recovery of fourier compressible functions. *SIAM Journal on Matrix Analysis and Applications*, 33(1):263–289, 2012.

[44] Xianfeng Hu, Mark Iwen, and Hyejin Kim. Rapidly computing sparse legendre expansions via sparse fourier transforms. *Numerical Algorithms*, pages 1–31, 2015.

[45] I.B. Segal and M.A. Iwen. Improved sparse fourier approximation results: Faster implementations and stronger guarantees. *Numerical Algorithms*, 63:239 – 263, 2013.

[46] Andrew Christlieb, David Lawlor, and Yang Wang. A multiscale sub-linear time fourier algorithm for noisy data. *Applied and Computational Harmonic Analysis*, 40:553 – 574, 2016.

[47] A. Gilbert, Y. Li, E. Porat, and M. Strauss. Approximate sparse recovery: Optimizing time and measurements. *SIAM Journal on Computing*, 41(2):436–453, 2012.

[48] Mark A Iwen. Compressed sensing with sparse binary matrices: Instance optimal error guarantees in near-optimal time. *Journal of Complexity*, 30(1):1–15, 2014.

[49] Anna C. Gilbert, Yi Li, Ely Porat, and Martin J. Strauss. For-all sparse recovery in near-optimal time. *ACM Trans. Algorithms*, 13(3):32:1–32:26, March 2017.

[50] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

[51] Deanna Needell and Roman Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *IEEE Journal of selected topics in signal processing*, 4(2):310–316, 2010.

[52] Tong Zhang. Sparse recovery with orthogonal matching pursuit under rip. *IEEE Transactions on Information Theory*, 57(9):6215–6221, 2011.

[53] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv:1011.3027v7*, 2011.

[54] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[55] R Arratia and L Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131, 1989.

[56] Leslie Greengard and June-Yub Lee. Accelerating the nonuniform fast fourier transform. *SIAM review*, 46(3):443–454, 2004.