PULSE: Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation

Sameer Pawar

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-215
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-215.html

December 17, 2013

PULSE: Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation

by

Sameer Anandrao Pawar

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engneering-Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor David Tse
Professor Michael Lustig
Professor John Lott

Fall 2013

PULSE: Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation

Abstract

PULSE: Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation

by

Sameer Anandrao Pawar

Doctor of Philosophy in Engneering-Electrical Engineering and Computer Science

University of California, Berkeley

Professor Kannan Ramchandran, Chair

An emerging challenge in recent years for engineers, researchers, and data scientists across the globe is to acquire, store, and analyze ever-increasing amounts of data. In the past decade, a new paradigm in data acquisition called "compressed-sensing" has emerged to cope with this data explosion. Compressed sensing exploits the fact that in many applications, although the signal of interest has a large ambient dimension, the relevant information resides in a significantly lower dimensional space. For example, the Magnetic-Resonance-Imaging (MRI) data is sparse in the wavelet-domain. In this thesis, we consider the problem of computing a sparse Discrete-Fourier-Transform of a high-dimensional signal from its time-domain samples, as a representative example of compressed-sensing problems. We use this problem to investigate the *tradeoff between the number of measurements, noise robustness, and the computational complexity of the recovery algorithm in compressed sensing problems.*

We propose a new family of *deterministic sparse sensing matrices*, obtained by blending together diverse ideas from sparse graph codes, digital signal processing, and number-theoretic concepts like the Chinese-remainder-theorem (CRT). The specific sparse structure of the proposed family of measurement matrices further enables a Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation, that are accordingly dubbed PULSE algorithms. Further, using the CRT, we establish an intimate connection between the problem of computing a sparse DFT of a signal and decoding over an appropriately designed sparse graph code. This connection is then exploited 1) to design a *sample efficient* measurement matrix and a *low-complexity* peeling-style iterative recovery algorithm, and 2) to perform a rigorous analysis of the recovery algorithm by wielding powerful and well-established analytical tools like *density-evolution*, *martingales*, and *expander graphs* from the coding theory literature. In particular, we show that under some mild conditions a k-sparse n-length DFT of a signal can be computed using (nearly optimal) $4k$ measurements and $O(k \log k)$ computations. As a concrete example, when $k = 300$, and $n = 3.8 \times 10^6$, our algorithm achieves computational savings by a factor of more than 6000, and savings in the number of input samples by a factor of more than 3900 over the standard Fast-Fourier-Transform

(FFT) algorithms. This can be a significant advantage in many existing applications and can enable new classes of applications that were not thought to be practical so far. Next, we extend these results to the case of noise-corrupted samples, computing sparse $2D$-DFTs as well as to interpolation of multi-variate sparse polynomials over the complex field and finite fields. We also demonstrate an application of the proposed PULSE algorithm to acquire the Magnetic-Resonance-Imaging of the Brain. This provides some empirical evidence that PULSE algorithms are applicable for acquiring more realistic signals. The proposed sensing framework and the recovery algorithm are sample efficient and robust against observation noise. We believe that this framework provides a possible direction towards designing efficient and low-power engineering solutions for sparse signal acquisition.

To my dear parents Anandrao and Sindhu,
and
my beloved wife Sushmita.

Contents

Acknowledgments

I would like to express my heart-felt gratitude to my advisor Prof. Kannan Ramchandran for his support and guidance that has aided this thesis in countless ways. I would also like to thank all the professors at Berkeley for offering amazing courses that taught me the fundamentals which helped me throughout my PhD program. I am also thankful to my thesis committee members Prof. David Tse, Prof. Michael Lustig, and Prof. John Lott.

I would like to thank my fellow graduate students and postdoctoral colleagues in the Wireless Foundations lab. They were an integral part of my life at Berkeley. They were friends, family, collaborators, teachers, and even philosophers depending on what I needed the most at that particular time. I will always cherish the time I spent with them for the rest of my life.

I am deeply indebted to my parents for their unconditional love and support throughout my life, for having faith in me and for encouraging me to pursue my dreams. No words can describe the support, devotion, encouragement and love I got from my lovely and beautiful wife Sushmita, during these last 6 years. Without my family's support and encouragement, it would not have been possible for me to finish this thesis.

Chapter 1

Introduction

An emerging challenge in recent years for engineers, researchers and data scientists across the globe is to acquire, store and analyze ever-increasing amounts of data. The sheer volume of data amassed by many applications, including genomics, meteorology, medical imaging, astronomy, network monitoring, etc., is rendering infeasible the traditional paradigm of acquiring the entire data and compressing it for subsequent storage or analysis. Fortunately, in many applications of interest, although the ambient dimension of the data to be acquired is large, the relevant information resides in a significantly lower dimensional space, i.e., signals are transform-domain sparse. For example, the Magnetic-Resonance-Imaging (MRI) data is sparse in the wavelet-domain. Researchers and engineers in the past have exploited this information to "compress" the data after acquisition, e.g., JPEG, MPEG, MP3, etc.

In the past decade, a new paradigm in data acquisition called "compressed-sensing" has emerged. Compressed sensing is a signal processing technique that acquires multiple linear combinations of a signal and reconstructs the signal by finding a solution to an underdetermined linear system, thus, effectively performing *compression at the time of acquisition*. The theoretical and algorithmic advances in compressed-sensing have impacted many real-world applications like Magnetic-Resonance-Imaging (MRI), crystallography, face-recognition, phase retrieval, network tomography, etc. The idea of *compression at acquisition*, however, is relatively old. The most simple and yet powerful way of achieving compression at acquisition is to sample the data at rates lower than the Nyquist-Shannon sampling theorem. In general this task is impossible if no additional structure other than bandlimitedness is imposed on the signal. However, if the signal is known to have some additional structure, one can acquire the data at a significantly lower rate. For example, consider a problem of acquiring a complex single sinusoidal signal $x(t) = Ae^{\imath\omega t+\phi}$, of an unknown amplitude A and an unknown frequency ω. Then, using only two samples, $x(0) = Ae^{\phi}$ and $x(T) = Ae^{\imath\omega T+\phi}$, we can determine the frequency $\omega = (1/T)\angle x(T)x^{\dagger}(0)$ and the amplitude $A = |x(0)|$. Parametric models for signal acquisition have been exploited in the field of statistical signal processing for a long time, e.g., MUSIC [71], ESPRIT [69], etc.

In recent years, the concept of sampling has been generalized to acquire *linear combina-*

tion of samples of the signal, i.e., *compressed-sensing*. The major algorithmic and theoretical developments in compressed-sensing can be mainly classified into 3 groups of techniques: *convex optimization based techniques*, *greedy-iterative algorithms* and *coding-theory based techniques*.

- **Convex optimization based techniques**: The problem of recovering a high - dimensional signal, that has a sparse structure in some transform-domain, from linear measurements, can be formulated as an "ℓ_0-norm" minimization problem, which is an instance of a non-convex optimization problem. Convex optimization techniques first relax the original combinatorial problem to a convex problem of minimizing an "ℓ_1-norm" and then design computationally efficient algorithms to solve this relaxed problem [22, 11, 8, 10, 9, 20, 7, 50, 25, 13, 78]. Later, these techniques identify sufficient conditions on the measurement matrix to guarantee a high-fidelity solution to the original problem, using the aforementioned convex relaxation. The bulk of the literature concentrates on designing either random or deterministic measurement matrices that satisfy some form of the so-called Restricted Isometry Property (RIP) [12], that further provides good reconstruction guarantees for the ℓ_1 minimization techniques, e.g., see [8, 10, 9, 20]. The RIP essentially characterizes matrices which are nearly orthonormal or unitary, when operating on sparse vectors. The convex optimization approach is known to provide a high level of robustness against both the measurement noise and observation noise.

- **Greedy iterative algorithms**: Greedy iterative algorithms attempt to solve the original ℓ_0-minimization problem directly using heuristics. For example, the classical Orthogonal Matching Pursuit (OMP) [61, 19, 75] builds a solution to the ℓ_0-minimization problem by finding one term of the sparse vector at a time by selecting at each step the column of the measurement matrix that correlates most strongly with the residual signal. Examples of other greedy algorithms include[60, 18, 23].

- **Coding theory based techniques**: In coding theory based compressed-sensing, the approach is to first design a sensing matrix, e.g., using expander graphs, and then construct recovery algorithms targeted to that particular sensing matrix. Examples of coding theory based compressed-sensing are [79, 41, 3, 44, 14, 49, 1].

The convex-optimization techniques and the greedy algorithms rely on designing measurement matrices that have RIP-like properties. Although it is known that randomly generated matrices, e.g., Gaussian random matrices and Rademacher matrices, satisfy the RIP with high probability, there is no practical algorithm to verify if a given matrix satisfies the RIP. Using random matrices for compressed-sensing of real-world signals is difficult due to the physical constraints of the sensing mechanism, storage constraints of random entries as well as the computational constraints of matrix multiplication during the recovery process.

Most of the coding-theory based techniques use measurement matrices constructed via *expander-graphs* with high expansion properties. There are many existential results in the literature for constructing graphs that have high expansion properties, but very few explicit constructions of such expanders are available. Moreover, many of these algorithms are not robust against measurement or observation noise. In some cases, the recovery algorithm is applicable only for positive-valued sparse signals [14, 49]. There is also another line of work based on *sketching algorithms*, e.g., [15, 16, 17]. Again, these algorithms are not very noise tolerant and in practice work only for *very sparse* signals.

The aforementioned drawbacks of existing techniques motivate us to address the problem of sparse signal estimation from a fundamental perspective. In the compressed sensing problem, there seems to be a complex interplay between the measurement efficiency, noise robustness and the computational complexity of the recovery algorithm. This thesis attempts to understand this complex relation between different metrics from a fundamental perspective, while being realistic in the system modeling. In particular, we investigate the *tradeoff between the number of measurements, noise robustness and the computational complexity of the recovery algorithm, for a realistic sensing mechanism like partial Fourier measurements.*

In this thesis, we use the problem of computing a sparse Discrete-Fourier-Transform of a high-dimensional signal from its time-domain samples, as a representative example of compressed-sensing problems. The reason for doing so is two-fold: 1) signals with sparse spectrum abound in real-world applications, e.g., radar, MRI, multimedia, cognitive radio etc. and 2) it provides a concrete problem statement to illustrate the new ideas proposed in this thesis, which further can be applied to other compressed sensing applications. The main contributions of this thesis are:

- We propose a new family of *deterministic* sparse sensing matrices, obtained by blending together diverse ideas from sparse graph codes, a recent workhorse in coding theory, and the Discrete Fourier Transform (DFT), an old workhorse in digital signal processing. The resulting measurement matrices consist of a carefully chosen subset of rows of the DFT matrix, guided by the Chinese-remainder-theorem (CRT) [4], thus making the sensing mechanism more amenable to an actual implementation as compared to other randomized constructions, e.g., random Gaussian matrices.

- The special sparse structure of the proposed family of measurement matrices enables a low complexity peeling-style iterative recovery algorithms. Accordingly, we dub the family of recovery algorithms as PULSE, which stands for Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation.

- Using the CRT-guided sub-sampling operation, we establish an intimate connection between the problem of computing a sparse DFT of a signal and decoding over appropriately designed sparse graph codes. This connection is further harnessed to 1) design sample-efficient measurement matrices, 2) design a low-complexity peeling-style iterative recovery algorithm, and 3) rigorously analyze the proposed iterative recovery

algorithm using powerful and well-established analytical tools like *density-evolution*, *martingales*, and *expander graphs* from the coding theory literature, to obtain *nearly tight* bounds on the number of measurements required for a sparse signal estimation. Thus further paving a way for adoption of sophisticated techniques like *message-passing*, from the coding theory literature to the compressed sensing problems in a more systematic way.

- Random measurement matrices like Gaussian matrices exhibit the RIP with optimal sample scaling, but have limited use in practice. For example, Gaussian random measurement matrices are not applicable to problems such as computing a sparse DFT of a signal from its time-domain samples. To the best of our knowledge, the tightest available RIP analysis (see [66]) of a matrix consisting of random subset of rows of an $n \times n$ DFT matrix requires $O(k \log^3 k \log n)$ samples to stably recover a k-sparse n-dimensional signal. In contrast, we show that a stable recovery of a k-sparse n-dimensional signal is possible using $O(k \log^2 k \log n)$ samples.

To summarize, we augment the compressive-sensing literature by a *family of sample-efficient deterministic sparse measurement matrices and the associated family of low complexity PULSE recovery algorithms*. However, there are a few caveats that we would like to emphasize. First, our results are for asymptotic regimes of the sparsity k and the signal dimension n, where k is sub-linear in n. Secondly, we assume a stochastic model on the input signal; in particular, we assume that the k-sparse signal has a uniformly random support. Thus, our proposed PULSE algorithms trades off the sample and the computational complexity for asymptotically zero probability of failure guarantees in a non-adversarial sparsity setting.

The rest of the thesis is organized as follows:

Chapter 2 In this chapter, we consider an idealized setting of a discrete compressed-sensing problem with no additional noise. We propose a family of *sparse measurement matrices* and an associated *low complexity algorithm* that recovers an n-length discrete k-sparse signal from *nearly optimal* $2(1+\epsilon)k$ *noiseless linear observations, in an order-optimal* $O(k)$ *computational cost, with a high probability.* Our proposed family of sparse measurement matrices features a novel hybrid mix of the DFT, an old workhorse in digital signal processing, and LDPC codes, a recent workhorse in coding theory, to generate a linear measurement lens through which to perform compressive sensing (CS). This hybrid structure endows the resulting family of sparse measurement matrices with several useful properties that can be further exploited to design a fast, low complexity reconstruction algorithm that is also provably optimal (up to small constant multiple) in the number of measurements. At a high level, our approach combines signal processing concepts (the Discrete Fourier Transform) with coding-theoretic concepts (LDPC codes) in a novel way to formulate a fast SWIFT (Short-and-Wide-Iterative-Fast-Transform) algorithm, for the compressive sensing problem.

Chapter 3 In this chapter, we address the problem of computing an n-length k-sparse DFT of a signal from its time-domain samples. We propose a novel CRT based sub-sampling "front-end" architecture that transforms the problem of computing a sparse DFT of a signal to that of decoding over an appropriately designed sparse graph code. This connection is further exploited to design a low-complexity peeling-style iterative recovery algorithm called FFAST, that stands for Fast Fourier Aliasing-based Sparse Transform. We rigorously analyze the performance of the FFAST algorithm using powerful and well-established analytical tools from the coding theory literature, like *density-evolution*, *martingales*, and *expander graphs*. In particular, we show that an n-length k-sparse DFT can be computed using as few as $4k$ carefully chosen time-domain samples, in $O(k \log k)$ arithmetic computations.

Chapter 4 In this chapter, we extend the FFAST framework proposed in Chapter 3, for computing a k-sparse n-length DFT, to address the effects of imperfect measurements due to additional noise at the acquisition sensors, i.e., observation noise. In particular, we show that the FFAST framework acquires and reconstructs an n-length signal $\vec{x}$, whose DFT $\vec{X}$ has k non-zero coefficients, using $O(k \log^2 k \log n)$ noise-corrupted samples, in $O(n \log n \log^2 k)$ computations. This contrasts with the best known scaling of the number of partial Fourier measurements, i.e., $O(k \log^3 k \log n)$, required for the RIP [66], which further guarantees a stable recovery of an n-length k-sparse DFT of a signal from its noise-corrupted time-domain samples.

Chapter 5 In this chapter, we extend the results of Chapter 3 to compute the sparse $2D$-DFT's of $2D$ signals. The main contribution of this chapter is a $2D$-FFAST algorithm that computes a k-sparse $2D$-DFT of an $N_0 \times N_1$ signal, where the dimensions N_0, N_1 are either co-prime or equal and the non-zero DFT coefficients have a uniformly random support, using $O(k)$ samples and in $O(k \log k)$ computations. We apply our proposed $2D$-FFAST algorithm to acquire an MRI of the 'Brain' from its Fourier samples. Thus, we demonstrate the applicability of the FFAST architecture to acquire more realistic signals, that are approximately sparse and have a "non-uniform" (or clustered) support of the dominant DFT coefficients.

Chapter 6 In this chapter, we show that FFAST is a more general framework and can be applied to other applications of sparse signal recovery. In particular, we address the problem of interpolating a multi-variate sparse polynomial, over either the complex or a finite field, from its evaluations. The key idea is to obtain evaluations of the underlying unknown polynomial over the roots of unity and transform the problem of sparse polynomial interpolation to that of computing a sparse large (degree of the polynomial) dimension DFT of the evaluation points. We can then use the algorithm proposed in Chapter 3 to compute a sparse high-dimensional DFT.

We conclude the thesis with a summary of results and some important future research directions in Chapter 7.

Chapter 2

Compressed sensing

2.1 Introduction

Compressed-sensing is a new paradigm for acquiring a sparse high-dimensional data. Recent advances in compressed-sensing have impacted many real-world applications like Magnetic-Resonance-Imaging (MRI), crystallography, face-recognition, phase retrieval, network tomography etc. In practice, many signals of interest are compressible or have *sparse* representation in some transform-domain, e.g., Fourier, wavelet, etc. In such cases, a small subset of the transform coefficients contain most or all of the signal energy, with most components being either zero or negligibly small. The goal of compressed-sensing is to acquire or sense and approximately reconstruct a high-dimensional sparse (or compressible) signal vector $\vec{x}$ from a significantly fewer number of linear measurements or samples, in a computationally efficient manner.

In this chapter, we propose a new family of *sparse measurement matrices* and an associated *low complexity algorithm* that recovers a high-dimensional discrete sparse signal from a *nearly-optimal number of noiseless linear observations in an order-optimal computational cost*. Our proposed family of matrices features a novel hybrid mix of the Discrete Fourier Transform (DFT), an old workhorse in digital signal processing, and Low Density Parity Check (LDPC) codes, a recent workhorse in coding theory, to generate a linear measurement lens through which to perform compressive sensing (CS). This hybrid structure endows the resulting family of sparse measurement matrices with several useful properties that can be exploited to design a fast, low complexity reconstruction algorithm that is also provably order optimal in the number of measurements. At a high level, our approach combines signal processing concepts (the Discrete Fourier Transform) with coding-theoretic concepts (LDPC codes) in a novel way to formulate a fast algorithm (dubbed the Short-and-Wide-Iterative-Fast-Transform or SWIFT algorithm) for the compressive sensing problem.

Although our framework applies in more general settings, in this chapter, in the interest of presentation clarity, we consider a discrete CS problem with no additional noise (or

approximately sparse signal). Later in Chapter 4, we discuss how the framework and the reconstruction algorithm can be modified to make it *robust* against observation noise.

In particular, we provide explicit constructions of the sparse measurement matrices and the associated reconstruction algorithm SWIFT, to acquire a discrete n-length signal $\vec{x}$ that has at most k non-zero coefficients. The SWIFT algorithm recovers the signal $\vec{x}$, with high probability, from $2k(1+\epsilon)$ number of linear measurements, for any $\epsilon > 0$, in $O(k)$ complex operations. This is the first[1] CS result that we are aware of which is *order optimal both in the number of the measurements and the computational complexity*. However, we hasten to add the caveat that this is attained by relaxing the signal sparsity model to be statistical (uniformly random) rather than worst-case adversarial as assumed in many of CS works.

The rest of the chapter is organized as follows: In Section 2.2, we introduce the signal model and formulate the problem. In Section 2.3, we provide a quick summary of the main results and contrast it with the state of the art results in literature. Section 2.4 provides explicit constructions of the family of sparse measurement matrices and a description of the SWIFT recovery algorithm.

2.2 System Model and Problem Formulation

The goal of compressed sensing is to acquire/sense and approximately reconstruct a high-dimensional sparse (or compressible) signal vector $\vec{x}$ from a significantly fewer number of linear measurements/samples. In an ideal setup of the problem formulation, the signal vector $\vec{x} \in \mathbb{C}^n$, is assumed to be exactly k-sparse ($k < n$), i.e., has at most k non-zero entries, and the task is to efficiently reconstruct the signal $\vec{x}$ from an under-determined system of linear measurements. More formally, recover the n-length k-sparse signal $\vec{x}$, from $m < n$ linear measurements (also see Fig. 2.1), obtained as follows:

$$\vec{y} = \boldsymbol{A}\vec{x}, \tag{2.1}$$

where $\boldsymbol{A} \in \mathbb{C}^{m \times n}$, is the *measurement matrix*. Further we make the following assumptions on the signal model

- k and n are both asymptotic.
- the support of non-zero values of $\vec{x}$ is uniformly random.
- amplitudes of the non-zero values of $\vec{x}$ are complex and drawn i.i.d from some continuous distribution.

Our goal is to design both, a family of sparse measurement matrices $\boldsymbol{A}$, and the associated reconstruction algorithm, such that the k-sparse n-dimensional signal $\vec{x}$ can be recovered from almost optimal ($O(k)$) number of measurements with minimal ($O(k)$) computational cost.

[1] An independently done contemporaneous work by M. Bakshi et al. [1], has some similarities, as well as several differences with the proposed algorithm in this chapter.

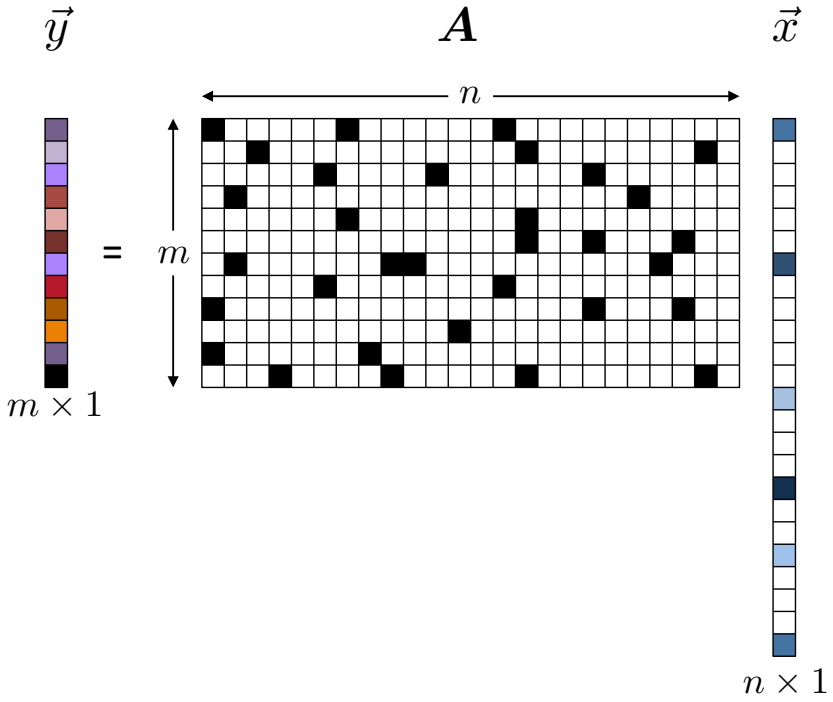


Figure 2.1: An n-dimensional k-sparse signal $\vec{x}$ is sensed using a sparse measurement $\boldsymbol{A}$ to get m linear measurements, as $\vec{y} = \boldsymbol{A}\vec{x}$.

2.3 Main Result

We propose a new family of sparse measurement matrices, constructed using a novel hybrid mix of the DFT matrices and LDPC codes, through which to perform compressive sensing (CS) of sparse high-dimensional discrete signals. This hybrid structure endows the resulting family of sparse measurement matrices with several useful properties that can be exploited to design a fast, low complexity (linear in k) iterative reconstruction algorithm which is also provably optimal (up to small constant multiple of k) in the number of measurements. The reconstruction algorithm is called SWIFT and stands for the Short-and-Wide-Iterative-Fast-Transform. The main result is summarized in the following theorem:

Theorem 2.3.1. *For any given sufficiently large values of k, n, we provide an explicit construction of a measurement matrix $\boldsymbol{A}$ and the associated reconstruction algorithm SWIFT, such that, given linear observations $\vec{y} = \boldsymbol{A}\vec{x}$, where $\vec{x}$ is an n-length k-sparse signal with a uniformly random support of the non-zero coefficients, the SWIFT-algorithm perfectly recovers the signal $\vec{x}$ with probability at least $1 - O(k^{-3/2})$. The SWIFT algorithm performs recovery using $m = 2k(1+\epsilon)$ number of measurements, for any $\epsilon > 0$, and $O(k)$ complex operations.*

Proof. Please see Appendix 2.A. □

2.3.1 Related work

The major algorithmic and theoretical developments in the field of compressed sensing can be broadly categorized into three major groups of techniques, *convex optimization based techniques*, *greedy-iterative algorithms* and *coding-theory based techniques*.

- **Convex optimization based techniques**: The problem of recovering a high dimensional signal, that has a sparse structure in some transform-domain, from linear measurements, can be formulated as an "ℓ_0-norm" minimization problem, which is an instance of a non-convex optimization problem. Convex optimization techniques first relax the original combinatorial problem to a convex problem of minimizing an "ℓ_1-norm" and then design computationally efficient algorithms to solve this relaxed problem [22, 11, 8, 10, 9, 20, 7, 50, 25, 13, 78]. Later, these techniques identify sufficient conditions on the measurement matrix to guarantee a high-fidelity solution to the original problem, using the aforementioned convex relaxation. The bulk of the literature concentrates on designing random measurement matrices that are characterized by $O(k\log(n/k))$ number of measurements and $O(poly(n))$ decoding complexity. The convex optimization approach is known to provide a high level of robustness against both the measurement noise and observation noise.

- **Greedy iterative algorithms**: Greedy iterative algorithms attempt to solve the original ℓ_0-minimization problem directly using heuristics. For example, the classical Orthogonal Matching Pursuit (OMP) [61, 19, 75] builds a solution to the ℓ_0-minimization problem by finding one term, of the sparse vector, at a time by selecting, at each step, the column of the measurement matrix that correlates most strongly with the residual signal. Examples of other greedy algorithms include [60, 18, 23]. Although greedy-algorithms are faster than the convex optimization based techniques, in order sense they are still $O(poly(n))$ in decoding complexity and require $O(k\log(n/k))$ number of measurements.

- **Coding theory based techniques**: In coding theory based compressed-sensing, the approach is to first design a sensing matrix, e.g., using expander graphs, and then construct recovery algorithms targeted to that particular sensing matrix. Examples of coding theory based compressed-sensing are [79, 41, 3, 44, 14, 49, 1]. While the measurement and the computational complexity of these algorithms is sub-linear, i.e, $O(k\log(n))$, it is still dependent on the ambient dimension n.

Thus, none of the existing CS solutions in the literature overcome the **"n-barrier"** w.r.t either the decoding complexity or the number of measurements, even for the noiseless regime. The Finite Rate of Innovations (FRI)-based techniques for signal acquisition [76], [21] and

[5], do overcome the "n-barrier" w.r.t the number of measurements for the noiseless regime, but not w.r.t. the decoding complexity, as it relies on computationally expensive polynomial root-finding operations, whose complexity is $O(k^2 \log(n))$ [2].

The main contribution of this chapter is to show that under no noise model i.e., when observations have no additional noise, it is possible to overcome the "n-barrier" for both, *the number of measurements and the decoding complexity*. However, we hasten to add the caveat that this is attained by relaxing the signal sparsity model to be statistical (uniformly random) rather than worst-case adversarial as assumed in many of CS works. Further, by carefully designing a hybrid DFT-LDPC measurement matrix family, we provide a fast "onion-peeling"-like SWIFT recovery algorithm that, with near-optimal number of measurements, reels off the uncovered non-zero entries of the sparse vector $\vec{x}$ in an order optimal $O(k)$ complex operations.

2.4 Design of Measurement matrix and associated recovery algorithm

In this section, we describe the construction of the sparse measurement matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$, and the associated fast iterative recovery algorithm SWIFT. Before we delve into the details of the measurement matrix design and the recovery algorithm, it is instructive to take a look at a "genie-assisted-peeling-decoder" (described below) over a carefully designed sparse bipartite graph. Our genie-assisted-peeling-decoder is inspired by the decoder used in [26] for "private stream search".

2.4.1 Genie-assisted peeling decoder over a sparse bipartite graph

We use a simple example to illustrate the genie-assisted-decoder. Consider an $n = 20$ length discrete signal $\vec{x}$ that has only $k = 5$ non-zero coefficients. Let the non-zero coefficients of $\vec{x}$ be $x[1] = 1$, $x[3] = 4$, $x[5] = 2$, $x[10] = 3$, $x[13] = 7$, while the rest of the coefficients are zero. We construct a bi-partite graph, with $n = 20$ variable nodes on the left and $n_b = 9$ parity check nodes (also referred as bins) on the right, as shown in Fig. 2.2.

Now, we assign the values of the left variable nodes in the graph of Fig. 2.2, to be equal to the entries of the k-sparse signal vector $\vec{x}$, i.e., at most k left nodes have non-zero values. Further, the values of the parity check nodes (right nodes of the bi-partite graph) are assigned to be equal to the complex sum of the values of its left neighbors. Since the zero-valued variable nodes do not affect the values of the parity check nodes, it suffices to show the graph connections only for the non-zero variable nodes (as shown in Fig. 2.2). The resulting bi-partite graph is *sparse*. A parity check node that has *exactly one* non-zero valued variable node, as a left neighbor, is called a 'single-ton' (green colored check nodes). Similarly, a parity check node that has *more than one* non-zero valued variable nodes, as left neighbors, is called a 'multi-ton' (red colored check nodes) and the one with all the zero-valued left

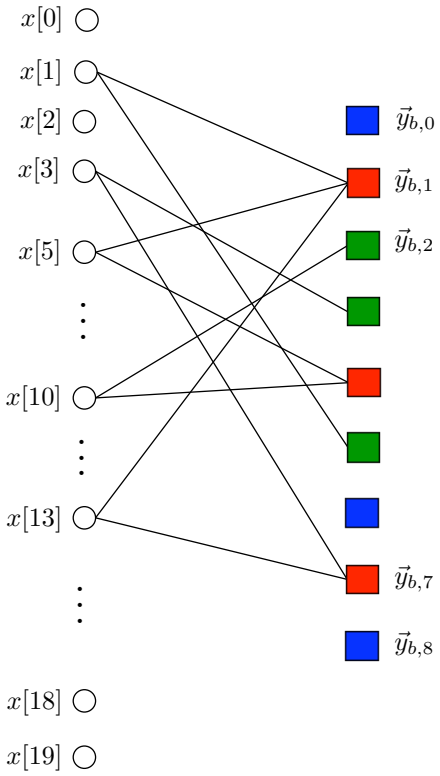


Figure 2.2: An example bipartite graph with $n = 20$ variable nodes on left and $n_b = 9$ parity check nodes (or bins) on right. The values of the variable nodes are set to be equal to the entries of the $n = 20$ length $k = 5$ sparse example signal $\vec{x}$. The value of a parity check node is equal to the complex sum of the values of its left neighbors. The observation of the parity check node i is denoted by $\vec{y}_{b,i}$. Since the zero-valued variable nodes do not affect the values of the parity check nodes, it suffices to show the graph connections only for the non-zero variable nodes. A parity check node that has *exactly one* non-zero valued variable node, as a left neighbor, is called a 'single-ton' (green colored check nodes). Similarly, a parity check node that has *more than one* non-zero valued variable nodes, as left neighbors, is called a 'multi-ton' (red colored check nodes) and the one with all the zero-valued left neighbors is called a 'zero-ton' (blue colored check nodes).

neighbors is called a 'zero-ton' (blue colored check nodes). Consider a 'genie' that informs the decoder *which* parity check nodes are zero-ton, single-ton and multi-ton. Further, for all the single-ton check nodes, it also informs the *support* of its only non-zero valued neighbor. Then, a decoder can identify singleton check nodes, and their support, with the help of the genie. Also, by observing the value of a single-ton parity check node, the decoder can infer the value of the neighboring variable node. Then, the genie-assisted-peeling decoder repeats the following steps:

1. Select all the edges in the graph with right degree 1 (edges connected to single-tons).
2. Remove these edges from the graph as well as associated left and right nodes.
3. Remove all other edges that were connected to the left nodes removed in step-2. When

a neighboring edge of any right node is removed, its contribution is subtracted from that parity check node.

If, at the end, all the edges have been removed from the graph, the genie-assisted-peeling-decoder would have successfully reconstructed the signal $\vec{x}$. For example, a genie-assisted-peeling decoding on graph in Fig. 2.2 results in a successful decoding, with the non-zero coefficients of the signal $\vec{x}$ recovered in the order $x[10], x[3], x[5], x[1]$ and $x[13]$. The probability of success of the genie-assisted-peeling decoder clearly depends on the properties of the sparse bipartite graph used for decoding.

In [52, 53], the authors have addressed this problem of designing optimal irregular edge degree sparse bi-partite graphs in the context of *erasure correcting codes*. Further, the authors of [53] have shown, an appropriate choice of the left edge degree distribution (to a truncated enhanced harmonic degree distribution) of the bipartite graph, the genie-assisted-peeling-decoder described above successfully recovers the signal $\vec{x}$ with probability at least $1 - O(k^{-3/2})$, given that the number of the check nodes $n_b \geq (1+\epsilon)k$ for any $\epsilon > 0$.

Thus, if the bi-partite graph is constructed using the appropriately designed LDPC matrix $\boldsymbol{H}$, such that the j^{th} left variable node is connected to the i^{th} right parity check node iff the entry $H[i][j] = 1$, then the peeling-decoder successfully recovers all the non-zero values of $\vec{x}$ with high probability under the following assumptions:

1. Zero-ton, single-ton and multi-ton bins are identified correctly.
2. If a bin is a single-ton, then the decoder also correctly identifies the support and the value of the contributing non-zero neighbor.

Next, we show how these properties/assumptions can be achieved with high probability without the help of the genie. The key idea is to have a 2-dimensional vector observation at each parity check node rather than a scalar observation.

Getting rid of the genie Again consider our $n = 20$ length $k = 5$ sparse example signal $\vec{x}$ and the bi-partite graph of Fig. 2.2. Now suppose, each variable node contributes a 2-dimensional complex vector, rather than a scalar complex number, to all the parity check nodes it is connected to on right. In particular, a variable node $x[\ell]$ contributes $(x[\ell], x[\ell]e^{\imath 2\pi\ell/n})$ to its neighboring check nodes. The observation vector $\vec{y}_b$ of a bin (or a check node) is a complex sum of all the 2-dimensional vectors from its left neighbors. For example, the observations of bins $0, 1$ and 2 in Fig. 2.2 are:

$$\begin{aligned}
\vec{y}_{b,0} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
\vec{y}_{b,1} &= x[1]\begin{pmatrix} 1 \\ e^{\imath 2\pi/20} \end{pmatrix} + x[5]\begin{pmatrix} 1 \\ e^{\imath 2\pi 5/20} \end{pmatrix} + x[13]\begin{pmatrix} 1 \\ e^{\imath 2\pi 13/20} \end{pmatrix} \\
\vec{y}_{b,2} &= x[10]\begin{pmatrix} 1 \\ e^{\imath 2\pi 10/20} \end{pmatrix}
\end{aligned}$$

More generally, the 2-dimensional complex observation vector $\vec{y}_b$ of a bin b is given by,

$$\vec{y}_b = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\imath 2\pi 1/n} & e^{\imath 2\pi 2/n} & \cdots & e^{\imath 2\pi (n-1)/n} \end{pmatrix} \vec{x}_b, \tag{2.2}$$

where $\vec{x}_b \in \mathbb{C}^n$ consists of all non-zero coefficients of $\vec{x}$ that are connected to bin b and rest of the entries are zero. The 2-dimensional observation vector of a bin can be used to determined if a bin is a zero-ton, a single-ton or a multi-ton bin as follows:

- **Zero-ton:** A zero-ton bin is identified trivially. For example,
$$\vec{y}_{b,0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$
- **Single-ton:** Using the observation vector $\vec{y}_{b,2}$ of bin 2, which is a singleton-bin, we can determine the support and the value of the non-zero variable node connected to bin 2 as follows:
  - *support*:
  $$10 = \frac{20}{2\pi} \angle y_{b,2}[1] y_{b,2}^{\dagger}[0].$$
  - *value*:
  $$x[10] = y_{b,2}[0].$$
  Also note, for a single-ton bin $|y_{b,2}[0]| = |y_{b,2}[1]|$ and the support is an integer between 0 to 19. It is easy to verify that this property holds for all the singleton bins. We refer to the procedure of computing the support as a 'ratio-test' in the sequel.
- **Multi-ton:** Consider the observation vector $\vec{y}_{b,1}$ of bin 1, which is a multi-ton bin, given as,
$$\begin{aligned} \vec{y}_{b,1} &= x[1] \begin{pmatrix} 1 \\ e^{\imath 2\pi/20} \end{pmatrix} + x[5] \begin{pmatrix} 1 \\ e^{\imath 2\pi 5/20} \end{pmatrix} + x[13] \begin{pmatrix} 1 \\ e^{\imath 2\pi 13/20} \end{pmatrix} \\ &= \begin{pmatrix} 10 \\ -3.1634 - \imath 3.3541 \end{pmatrix} \end{aligned}$$
Now, if we perform the 'ratio-test' on the observation vector $\vec{y}_{b,1}$, we get the support to be 12.59. Also, $|y_{b,1}[0]| \neq |y_{b,1}[1]|$. Since, we know that for a singleton bin the magnitudes of both observations are identical and the support is an integer value between 0 and 19, we conclude that the observations $\vec{y}_{b,1}$ correspond to a multi-ton bin. In Appendix 2.A, we rigorously show that the ratio-test successfully identifies a multi-ton bin almost surely.

Thus, using a 2-dimensional complex observation vector at each bin and the ratio-test, the peeling-decoder can successfully identify zero-ton, single-ton, multi-ton bins *without the help of the genie*, with high probability.

2.4.2 Measurement Matrix $\boldsymbol{A}$

In this section, we describe the construction of a family of sparse measurement matrices $\{\boldsymbol{A}(n,k)\}$ indexed by the problem parameters n, k. For a given (n,k), the measurement matrix $\boldsymbol{A}$ is such that the observations $\vec{y}$ and the signal $\vec{x}$ are related through a peeling-friendly sparse bipartite graph (as shown in Fig. 2.2) and each bin has a 2-dimensional complex observation vector as discussed in Section 2.4.1. Thus, enabling us to use a peeling-decoder to successfully reconstruct $\vec{x}$ with high probability.

First, we define a matrix operator *row-tensor product* denoted as '$\otimes_r$' (also see [33]). Consider two matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ given as,

$$\boldsymbol{B}^T = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \end{pmatrix}, \text{ and } \boldsymbol{C}^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Then, a row-tensor product $\boldsymbol{B}^T \otimes_r \boldsymbol{C}^T$ is given by,

$$\boldsymbol{B}^T \otimes_r \boldsymbol{C}^T = \begin{pmatrix} 1 & 0 & 3 & 0 \\ 1 & 0 & 9 & 0 \\ 0 & 2 & 0 & 4 \\ 0 & 4 & 0 & 16 \\ 1 & 0 & 0 & 4 \\ 1 & 0 & 0 & 16 \end{pmatrix}.$$

In general if $\boldsymbol{B}^T \in \mathbb{F}^{p\times q}$ and $\boldsymbol{C}^T \in \mathbb{F}^{r\times q}$, for some arbitrary field $\mathbb{F}$, then the row-tensor product $\boldsymbol{B}^T \otimes_r \boldsymbol{C}^T \in \mathbb{F}^{pr\times q}$. The j^{th} set of p rows of $\boldsymbol{B}^T \otimes_r \boldsymbol{C}^T$ is given by $\{(\vec{c}_j \circ \vec{b}_i)^T, 0 \leq i \leq p-1\}$ for all $0 \leq j \leq r-1$, where $\vec{c}_j, \vec{b}_i$ are j^{th} and i^{th} rows of matrices $\boldsymbol{C}^T$ and $\boldsymbol{B}^T$ respectively and '$\circ$' denotes the Hadamard product of the rows.

Now, the measurement matrix $\boldsymbol{A}$ for a given (n,k) and $\epsilon > 0$ is,

$$\boxed{\boldsymbol{A} = \boldsymbol{F} \otimes_r \boldsymbol{H}} \tag{2.3}$$

where,

$$\boldsymbol{F} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\imath 2\pi 1/n} & e^{\imath 2\pi 2/n} & \cdots & e^{\imath 2\pi (n-1)/n} \end{pmatrix}, \tag{2.4}$$

consists of the first two rows of an $n \times n$ DFT matrix and $\boldsymbol{H} \in \mathbb{C}^{(1+\epsilon)k \times n}$ is a binary LDPC matrix with $\{0,1\}$ entries. Thus the total number of measurements is $2(1+\epsilon)k$. The LDPC matrix $\boldsymbol{H}$ is designed using an optimal irregular construction from [53].

Algorithm 1 SWIFT Algorithm

Input: The bin observations $\{\vec{y}_{b,i}\}_{i=0}^{n_b-1}$ obtained through the sparse measurement matrix $\boldsymbol{A}$.

Output: An estimate $\vec{x}$ of the n-length, k-sparse signal.

SWIFT Decoding: Set the initial estimate $\vec{x} = 0$. The SWIFT decoder processes the bin observations for a constant number of iterations, as follows:

```
for each iteration do
  for each bin i = 0 to n_b − 1 do
    if ||y_{b,i}||^2 == 0 then
      bin i is a zero-ton.
    else
      (singleton, v_p, p) = Singleton-Estimator (y_{b,i}).
      if singleton = 'true' then
        y = y − v_p A(:, p).
        Set, x[p] = v_p.
      else
        bin i is a multi-ton.
      end if
    end if
  end for
end for
```

2.4.3 Recovery algorithm: SWIFT

Our recovery algorithm dubbed Short-and-Wide Iterative Fast Transform or 'SWIFT' is an iterative algorithm along the lines of the 'genie-assisted-peeling-decoder' explained in Section 2.4.1 (see algorithm 1 for the pseudo-code). The key difference from the genie assisted algorithm is a novel way of detecting if a bin is a single-ton or a multi-ton bin using the function *SingletonEstimator()* (see algorithm 2 for the pseudo-code).

The SWIFT algorithm performs a constant number of iterations. In each iteration the SWIFT algorithm processes all the bins. For each bin, it first computes the energy of the observation vector to detect a zero-ton. Further, if a bin has non-zero energy, it uses the function *SingletonEstimator()* to distinguish between a singleton bin and a multi-ton bin. The function *SingletonEstimator()* sets the indicator flag 'singleton' to 1 if the processed bin is a singleton and to 0 if the processed bin is a multi-ton. In the case of a singleton bin the function *SingletonEstimator()* also provides the value and the support of the non-zero coefficient. The SWIFT algorithm then peels of the contribution of any singleton bins found during the current iteration, thus inducing more singletons to be processed in the next iteration.

Algorithm 2 Singleton-Estimator

1: *Input:* The bin observation $\vec{y}_{b,i}$.

2: *Outputs:* 1) An indicator flag 'singleton', 2) Estimate of the support p and the value v_p of the non-zero coefficient, if the bin is a singleton bin.

3: *Singleton-Estimator:*
4: Set singleton = 'false'.
5: *Ratio-test:* $p = (n/2\pi)\angle y_{b,i}[1]y^{\dagger}_{b,i}[0]$.
6: **if** $|y_{b,i}[0]| == |y_{b,i}[1]|$ and $p \in \{0, 1, \ldots, (n-1)\}$ **then**
7: singleton = 'true'.
8: $v_p = y_{b,i}[0]$.
9: **end if**

2.A Proof of Theorem 2.3.1

2.A.1 Probability of success

Following the discussion in Section 2.4, it suffices to show that the SWIFT algorithm satisfies the following properties with probability at least $1 - O(k^{-3/2})$,

1. zero-ton, single-ton and multi-ton bins are identified correctly.
2. If a bin is a singleton-bin, then the decoder also identifies the support and the value of the non-zero coefficient connected to that bin.

It is trivial to verify that if a bin is a zero-ton or a single-ton, it is always identified correctly. Also, in the case of a singleton bin, the estimated support and the value of the non-zero coefficient is correct.

Next, consider a multiton-bin ℓ with $L-1$ non-zero components where $L > 2$. Let $i_0, i_1, \cdots, i_{L-2}$ be the support of the non-zero components of $\vec{x}$ contributing to the observation of the multi-ton bin ℓ. Then, we have

$$\vec{y}_{b,\ell} = \begin{bmatrix} \vec{f}_{i_0} & \vec{f}_{i_1} & \cdots & \vec{f}_{i_{L-3}} & \vec{f}_{i_{L-2}} \end{bmatrix} \begin{bmatrix} x[i_0] \\ x[i_1] \\ \vdots \\ x[i_{L-3}] \\ x[i_{L-2}] \end{bmatrix}, \tag{2.5}$$

where $\vec{f_i}$ is the i^{th} column of the matrix $\boldsymbol{F}$, given in (2.4). The multi-ton bin ℓ is identified as a single-ton bin with support j and value $x[j]$ for some $0 \leq j < n$, iff,

$$\begin{aligned} \vec{y}_{b,\ell} &= \vec{f}_j x[j] \\ i.e., \begin{bmatrix} \vec{f}_{i_0} & \vec{f}_{i_1} & \cdots & \vec{f}_{i_{L-3}} \end{bmatrix} \begin{bmatrix} x[i_0] \\ x[i_1] \\ \vdots \\ x[i_{L-3}] \end{bmatrix} &= \begin{bmatrix} \vec{f}_{i_{L-2}} & \vec{f}_j \end{bmatrix} \begin{bmatrix} -x[i_{L-2}] \\ x[j] \end{bmatrix} \\ \vec{u} &= \begin{bmatrix} \vec{f}_{i_{L-2}} & \vec{f}_j \end{bmatrix} \begin{bmatrix} -x[i_{L-2}] \\ x[j] \end{bmatrix}. \end{aligned} \tag{2.6}$$

where $\vec{u} \in \mathbb{C}^2$, is some resultant vector. Since $\boldsymbol{F}$ is a Vandermonde matrix, equation (2.6) has a unique solution. Since the complex coefficient $x[i_{L-2}]$ is drawn from a continuous distribution, the probability of satisfying equation (2.6) is essentially zero. Also, the probability of a multi-ton bin having a zero energy is upper bounded by the probability of a multi-ton bin being identified as a singleton, which is essentially 0. Hence a multi-ton bin is identified correctly almost surely.

2.A.2 Sample complexity

In Section 2.4.2, we have shown that by using optimal irregular sparse graph constructions from [53], the SWIFT algorithm can reconstruct the sparse signal $\vec{x}$ with probability at least $1 - O(k^{-3/2})$, from $2(1+\epsilon)k$ samples for any $\epsilon > 0$.

2.A.3 Computational complexity

In [53], the authors have shown that the genie-assisted-peeling decoder successfully recovers the signal $\vec{x}$ in a constant number of iterations, i.e., $O(k)$ complex computations. The SWIFT algorithm has an additional function *SingletonEstimator()* whose computational complexity is $O(1)$. Hence, the SWIFT algorithm recovers all the non-zero coefficients of the signal $\vec{x}$ in $O(k)$ complex operations. ■

Chapter 3

Computing a sparse discrete-Fourier-transform

3.1 Introduction

Spectral analysis using the Discrete Fourier Transform (DFT) has been of universal importance in engineering and scientific applications for a long time. A direct computation of the DFT $\vec{X}$ of an n-length signal $\vec{x}$ from the definition is often too slow, i.e., $O(n^2)$[1], to be practical. As a result, researchers have developed a family of algorithms called Fast Fourier Transform (FFT) (see [4]) to compute the same result in more computationally efficient way i.e., $O(n \log n)$, arithmetical operations. In practice many applications of interest involve signals, e.g. relating to audio, image, and video data, seismic signals, biomedical signals, financial data, social graph data, cognitive radio applications, surveillance data, satellite imagery, etc., which have a *sparse* Fourier spectrum. In such cases, a small subset of the spectral components typically contains most or all of the signal energy, with most spectral components being either zero or negligibly small. For such signals, is it possible to exploit this additional knowledge to devise an algorithm that can compute the spectrum in a more efficient way than FFTs?

In this chapter we address this question and answer it affirmatively. In particular, we show that if an n-length signal $\vec{x}$ is known to have a k-sparse DFT, $\vec{X}$, where $k << n$, then, under certain additional conditions, our proposed algorithm computes the DFT using $O(k)$ carefully chosen input samples in $O(k \log k)$ arithmetical operations. The improvement in speed can be enormous, especially for long signals where n may be in the order of hundreds of thousands or millions. This can be a significant advantage in many existing applications,

[1]Recall that a single variable function $f(x)$ is said to be $O(g(x))$, if for a sufficiently large x the function $|f(x)|$ is bounded above by $|g(x)|$, i.e., $\lim_{x\to\infty} |f(x)| < c|g(x)|$ for some constant c. Similarly, $f(x) = \Omega(g(x))$ if $\lim_{x\to\infty} |f(x)| > c|g(x)|$ and $f(x) = o(g(x))$ if the growth rate of $|f(x)|$ as $x \to \infty$, is negligible as compared to that of $|g(x)|$, i.e. $\lim_{x\to\infty} |f(x)|/|g(x)| = 0$.

as well as enable new classes of applications not thought practical so far.

We emphasize the following caveats. First, our analytical results are probabilistic, and work asymptotically in k and n, where k is sub-linear in n, with a success probability that approaches 1 asymptotically. This contrasts the $O(n \log n)$ FFT algorithm which works deterministically for all values of k and n. Secondly, we assume that the support of the non-zero DFT coefficients is uniformly random. Lastly, we require the signal length n to be a product of a few (typically 3 to 9) distinct primes of same order[2].

In effect, our algorithm trades off the sample and the computational complexity for asymptotically zero probability of failure guarantees in a non-adversarial sparsity setting, and is applicable whenever k is sub-linear in n (i.e. k is $o(n)$), but is obviously most attractive when k is much smaller than n. As a concrete example, when $k = 300$, and $n = 2^7 3^5 5^3 \approx 3.8 \times 10^6$, our algorithm achieves computational savings by a factor of more than 6000, and savings in the number of input samples by a factor of more than 3900 over the standard FFT (see [73] for computational complexity of a prime factor FFT algorithm).

Main idea At a high level, our algorithm uses a small and structured set of uniform subsampling operations applied directly on the input signal. It is well known from basic sampling theory that subsampling without anti-alias filtering creates spectral aliasing that destroys the original signal spectrum. Our key idea is to exploit rather than avoid this aliasing. This is done by recognizing that a *carefully designed subsampling operation will induce spectral aliasing artifacts that look like the parity constraints of good erasure-correcting codes that additionally have a fast peeling-decoding algorithm* (see Section 3.4.3 for more details). The resulting algorithm is low in both the sample complexity and the computational complexity. We accordingly dub our algorithm the FFAST (Fast Fourier Aliasing-based Sparse Transform). Our main inspiration is drawn from the class of sparse graph-codes for erasure channels studied in coding theory, e.g., Low-Density-Parity-Check (LDPC) codes [27], fountain codes [6, 55], verification codes [56], etc. Why? These codes have two important properties that we would like to inherit: (a) they have very low computational complexity (iterative peeling-based) decoder; and (b) they are near-capacity achieving for the erasure channel. The first property bestows the desired low computational complexity, while the second property ensures that the sample complexity of the algorithm is near-optimal.

But how do we achieve this goal? We cannot induce any arbitrary code in the spectral-domain at our will as we can control only the subsampling operation on the time-domain signal. The key idea is to design subsampling patterns, guided by the *Chinese-Remainder-Theorem (CRT)*, that create the desired code-like aliasing patterns. As we will describe in Section 3.5, the subsampling patterns are based on relatively co-prime integers for the very-sparse regime (sparsity-index $0 < \delta \leq 1/3$). When the spectrum is less-sparse ($1/3 < \delta <$

[2]This is not a major restriction as in many problems of interest, the choice of n is available to the system designer, and choosing n to be a power of 2 is often invoked only to take advantage of the readily-available radix-2 FFT algorithms.

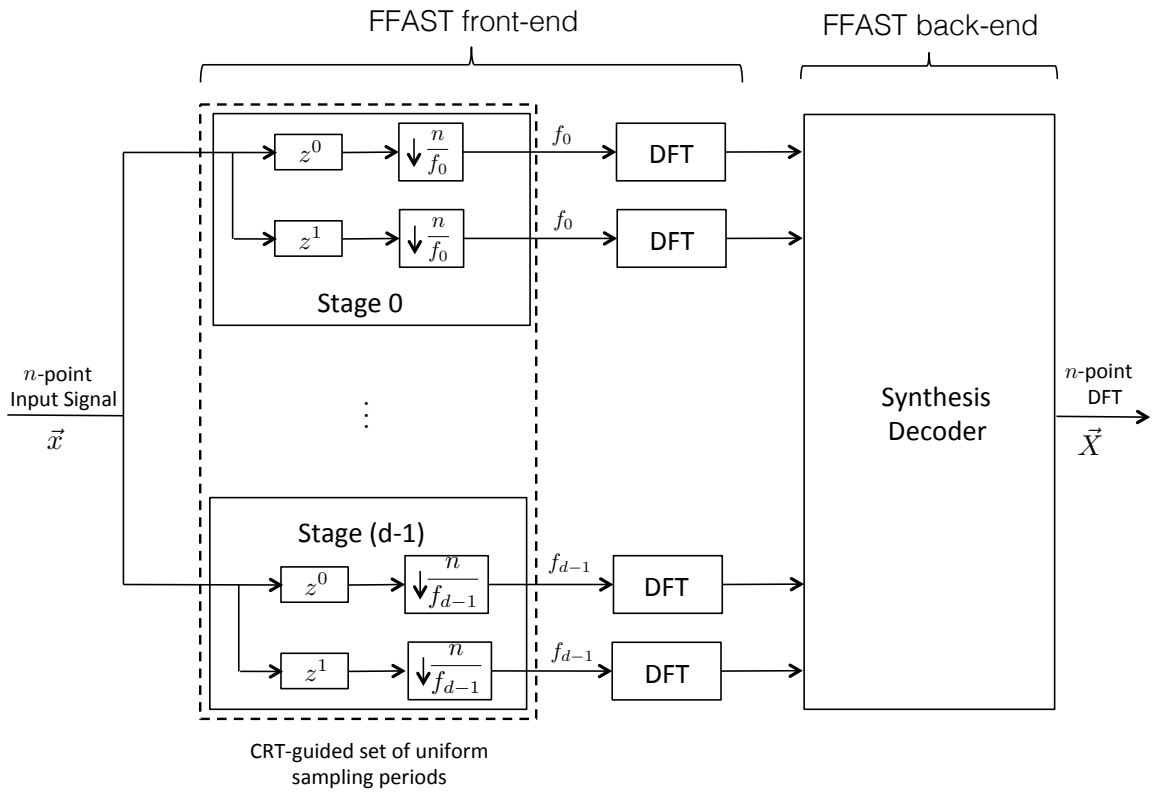


Figure 3.1: Schematic block diagram of the FFAST architecture. The n-point input signal $\vec{x}$ is uniformly subsampled by a carefully chosen set of d patterns, guided by the Chinese-Remainder-Theorem, to obtain $2d$ sub-streams. Each of the d stages has 2 sub-streams. A sub-stream is of length approximately equal to the sparsity k, resulting in an aggregate number of samples $m = 2\sum_{i=0}^{d-1} f_i \triangleq rk$, for a small constant r. Next, the (short) DFTs, of each of the resulting sub-streams are computed using an efficient FFT algorithm of choice. The big n-point DFT $\vec{X}$ is then synthesized from the smaller DFTs using the peeling-like FFAST decoder.

1), the subsampling patterns are a bit more complicated, comprising of "cyclically-shifted" overlapping co-prime integers. See Section 3.6 for details.

Our approach is summarized in Fig. 3.1. The n-point input signal $\vec{x}$ is uniformly subsampled through d stages. Each of the d stages, subsample the input signal and its circularly shifted version by a carefully chosen set of d patterns, guided by the Chinese-Remainder-Theorem, to obtain $2d$ sub-streams of sampled data. Each sub-stream is of length approximately equal to the sparsity k, resulting in an aggregate number of samples $m = 2\sum_{i=0}^{d-1} f_i \triangleq rk$, for a small constant r. Next, the (short) DFTs, of each of the resulting sub-streams are computed using an efficient FFT algorithm of choice. The big n-point DFT $\vec{X}$ is then synthesized from the smaller DFTs using the peeling-like FFAST decoder. As a concrete example, if the signal $\vec{x}$ is of length $n \approx 10^6$ and its DFT $\vec{X}$ has $k = 200$ non-zero coefficients, then the FFAST algorithm first generates 6 ($d = 3$) uniformly sampled sub-streams of the input data, each of size ≈ 100 ($f_i \approx 100,\ i = 0, 1, 2$), and then synthesizes $\vec{X}$

from the short DFTs of these 6 sub-streams. Due to the linear-time (in k) decoding complexity of the peeling-like FFAST decoder, our main computational bottleneck is in taking a small constant number of short (linear in k) DFTs. The precise number and the size of the short DFTs is quantified in Section 3.3. For the entire range of practical interest of sub-linear sparsity (i.e. $k \propto n^{\delta}$ where $0 < \delta < 0.99$), the overall sample complexity of the FFAST algorithm is no more than $4k$, and the computational complexity is $O(k \log k)$ with small constants in the big-Oh. This is particularly gratifying because both the sample complexity and the computational complexity depend only on the sparsity parameter k, which is sub-linear in n.

In this chapter we focus on the signals that have exactly k-sparse DFT $\vec{X}$ and there is no additional noise in the observations. Our motivation for this focused study is: (i) to provide conceptual clarity of our proposed approach in a noiseless setting; (ii) to present our deterministic subsampling front-end measurement subsystem as a viable alternative to the class of randomized measurement matrices popular in the compressive sensing literature [20, 8]; and (iii) to explore the fundamental limits on both sample complexity and computational complexity for an exact-sparse DFT, which is of intellectual interest. Later, in Chapter 4 we discuss the required modifications to make the FFAST algorithm robust against observation noise.

3.1.1 Related Work

A number of previous works [31, 32, 35, 38, 39] have addressed the problem of computing a 1-D DFT of a discrete-time signal that has a sparse Fourier spectrum, in sub-linear sample and time complexity. Most of these algorithms achieve a sub-linear time performance by first isolating the non-zero DFT coefficients into different bins, using specific filters or windows that have 'good' (concentrated) support in both, time and frequency. The non-zero DFT coefficients are then recovered iteratively, one at a time. The filters or windows used for the binning operation are typically of length $O(k \log n)$. As a result, the sample and computational complexity is typically $O(k \log n)$ or more. Moreover the constants involved in the big-Oh notation can be large, e.g., the empirical evaluation of [32] presented in [43] shows that for $n = 2^{22}$ and $k = 7000$, the number of samples required are $m \approx 2^{21} = 300k$ which is 75 times more than the sample complexity $4k$ of the FFAST algorithm[3]. The work of [35] provides an excellent tutorial on some of the key ideas used by most sub-linear time sparse FFT algorithms in the literature. In [42], the author proposes a sub-linear time algorithm with a sample complexity of $O(k \log^4 n)$ or $O(k^2 \log^4 n)$ and computational complexity of $O(k \log^5 n)$ or $O(k^2 \log^4 n)$ to compute a sparse DFT, with high probability or zero-error respectively. The algorithm in [42] exploits the Chinese-Remainder-Theorem, along with $O(poly(\log n))$ number of subsampling patterns to identify the locations of the non-zero

[3] As mentioned earlier, the FFAST algorithm requires the length of the signal n to be a product of a few distinct primes. Hence, the comparison is for an equivalent $n \approx 2^{22}$ and $k = 7000$.

DFT coefficients. In contrast, the FFAST algorithm exploits the CRT to induce 'good' sparse-graph codes using a small constant number of subsampling patterns and computes the sparse DFT with a vanishing probability of failure.

In summary, the FFAST algorithm is the first that we are aware of to compute an exactly k-sparse n-point DFT that has all of the following features:

- it has $O(k)$ sample complexity and $O(k \log k)$ computational complexity;
- it covers the *entire* sub-linear regime ($k \propto n^{\delta}, 0 < \delta < 1$);
- it has a probability of failure that vanishes to zero asymptotically;
- it features the novel use of the Chinese Remainder Theorem to guide the design of a small deterministic set of uniform subsampling patterns that induce good sparse-graph channel codes.

The rest of the chapter is organized as follows. In Section 3.2, we provide the problem statement along with the appropriate modeling assumptions and introduce the commonly used notations. Section 3.3 presents our main results and also provides a brief overview of the related literature. Section 3.4 exemplifies the mapping of computing the sparse DFT of a signal to decoding of an appropriately designed sparse-graph code. Sections 3.5 and 3.6 provide the performance analysis of the FFAST algorithm for the *very-sparse* and the *less-sparse* regimes respectively. In Section 3.8, we provide extensive simulation results that corroborate our theoretical findings, and validate the empirical performance of the FFAST algorithm.

3.2 Problem formulation, notations and preliminaries

3.2.1 Problem formulation

Consider an n-length discrete-time signal $\vec{x}$ that is a sum of $k << n$ complex exponentials, i.e., its n-length discrete Fourier transform has k non-zero coefficients:

$$x[p] = \sum_{q=0}^{k-1} a_q e^{2\pi \iota \omega_q p/n}, \qquad p = 0, 1, \ldots, n-1, \tag{3.1}$$

where the discrete frequencies $\omega_q \in \{0, 1, \ldots, n-1\}$ and the amplitudes $a_q \in \mathbb{C}$, for $q = 0, 1, \ldots, k-1$. We consider the problem of identifying the k unknown frequencies ω_q and the corresponding complex amplitudes a_q from the time-domain samples $\vec{x}$. A straightforward solution is to compute an n-length DFT, $\vec{X}$, using a standard FFT algorithm [4], and locate the k non-zero coefficients. Such an algorithm uses n samples and $O(n \log n)$ computations.

When the DFT $\vec{X}$ is known to be exactly k-sparse and $k << n$, computing all the n DFT coefficients seems redundant.

In this chapter, we address the problem of designing an algorithm, to compute the k-sparse n-point DFT of $\vec{x}$ for the asymptotic regime of k and n, when the support of the non-zero DFT coefficients is uniformly random. We would like the algorithm to have the following features:

- it takes as few input samples m of $\vec{x}$ as possible.
- it has a low computational cost that is a function of only the number of input samples m.
- it is applicable for the entire sub-linear regime, i.e., for all $0 < \delta < 1$, where $k = O(n^{\delta})$.
- it computes the DFT $\vec{X}$ with a probability of failure vanishing to 0 as m becomes large.

In the next section, we setup the notations and provide definitions of the important parameters used in the rest of the chapter.

3.2.2 Notation and preliminaries

Notation	Description
n	Ambient dimension of the signal $\vec{x}$.
k	Number of non-zero coefficients in the DFT $\vec{X}$.
δ	Sparsity-index: $k \propto n^{\delta}, 0 < \delta < 1$.
m	Sample complexity: Number of samples of $\vec{x}$ used by the FFAST algorithm to compute the DFT $\vec{X}$.
$r = m/k$	Oversampling ratio: Number of samples per non-zero DFT coefficient.
d	Number of stages in the "sub-sampling front end" of the FFAST architecture.
f_i	Number of samples of $\vec{x}$ per sub-stream, in the i^{th} stage of the "sub-sampling front end" of the FFAST architecture; $m = 2\sum_{i=0}^{d-1} f_i$.

Table 3.1: Glossary of important notations and definitions used in the rest of the chapter. The last three parameters are related to the "sub-sampling front end" of the proposed FFAST architecture (see Figure 3.1 for details).

Modulo-operator: For integers a, N, we use $(a)_N$ to denote the operation, $a \mod N$, i.e., $(a)_N \triangleq a \mod N$.

We now describe the Chinese-Remainder-Theorem (CRT) which plays an important role in our proposed FFAST architecture as well as in the FFAST decoder.

Theorem 3.2.1 (Chinese-Remainder-Theorem [4])**.** *Suppose $n_0, n_1, \ldots, n_{d-1}$ are pairwise co-prime positive integers and $N = \prod_{i=0}^{d-1} n_i$. Then, every integer 'a' between 0 and $N-1$ is uniquely represented by the sequence $r_0, r_1, \ldots, r_{d-1}$ of its remainders modulo $n_0, \ldots, n_{d-1}$ respectively and vice-versa.*

Further, given a sequence of remainders $r_0, r_1, \ldots, r_{d-1}$, where $0 \leq r_i < n_i$, Gauss's algorithm can be used to find an integer 'a', such that,

$$(a)_{n_i} \equiv r_i \quad \text{for } i = 0, 1, \ldots, d-1. \tag{3.2}$$

For example, consider the following pairwise co-prime integers $n_0 = 3, n_1 = 4$ and $n_2 = 5$. Then, given a sequence of remainders $r_0 = 2, r_1 = 2, r_2 = 3$, there exists a unique integer 'a', such that,

$$\begin{aligned} 2 &\equiv a \mod 3 \\ 2 &\equiv a \mod 4 \\ 3 &\equiv a \mod 5 \end{aligned} \tag{3.3}$$

It is easy to verify that $a = 38$ satisfies the congruencies in (3.3). Further, $a = 38$ is a unique integer modulo $N = n_0 n_1 n_2 = 60$ that satisfies (3.3).

3.3 Main Results

We propose a novel FFAST algorithm to compute the (exactly) k-sparse n-point DFT, $\vec{X}$, of an n-point signal $\vec{x}$. The n-length input signal $\vec{x}$ is such that its DFT $\vec{X}$ has *at most* k non-zero coefficients, i.e., $||\vec{X}||_0 \leq k$, with arbitrary complex values and uniformly random support in $\{0, 1, \ldots, n-1\}$. The FFAST algorithm computes the k-sparse n-point DFT with a high probability, using as few as $O(k)$ samples of $\vec{x}$ and $O(k \log k)$ arithmetic computations. The following theorem states the main result:

Theorem 3.3.1. *For any given $0 < \varepsilon < 1$, there exist (infinitely many) sufficiently large n, such that the FFAST algorithm computes the k-sparse DFT $\vec{X}$, where $k = \Omega(n^\delta)$ and $0 < \delta < 1$, of an n-length input signal $\vec{x}$, with the following properties:*

1. **Sample complexity:** *The algorithm needs $m = r(\delta)k$ samples of the signal $\vec{x}$, where the oversampling ratio $r(\delta) > 1$, is a small constant that depends on the sparsity index δ;*

2. **Computational complexity:** *The computational complexity of the FFAST algorithm is $O(k \log(k))$, where the constant in big-Oh is small.*

3. **Probability of success:** *The FFAST algorithm successfully computes all the non-zero DFT coefficients of the signal $\vec{x}$, with probability at least $1 - \varepsilon$.*

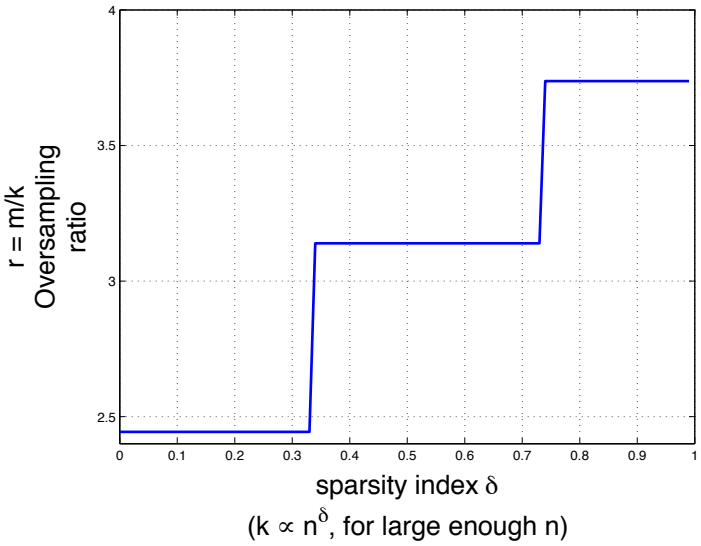


Figure 3.2: The plot shows the relation between the oversampling ratio $r = m/k$, and the sparsity index δ for $0 < \delta < 0.99$, where $k \propto n^{\delta}$. The FFAST algorithm successfully computes the k-sparse n-point DFT $\vec{X}$ of the desired n-point signal $\vec{x}$ with high probability, as long as the number of samples m is above the threshold given in the plot. Note that for nearly the entire sub-linear regime of practical interest, e.g. $k < n^{0.99}$, the oversampling ratio $r < 4$. For asymptotic values of k, the oversampling ratio $r = 2d\eta$, where d is the number of stages in the FFAST architecture and η is the average number of samples per sub-stream normalized by the number of non-zero coefficients k. In Section 3.5.4, we show that the number of stages d used in the FFAST architecture increases as δ approaches 1. The above achievable threshold plot is then obtained by using the constructions in Section 3.5.4 and the values of $d\eta$ from Table 3.4 in Section 3.5.3.

Proof. We prove the theorem in three parts. In Section 3.5, we analyze the performance of the FFAST algorithm for the very-sparse regime ($0 < \delta \leq 1/3$), and in Section 3.6 we analyze the less-sparse regime $1/3 < \delta < 1$. Lastly, in Section 3.7 we analyze the sample and the computational complexity of the FFAST algorithm. □

Note, that although Theorem 3.3.1 is for asymptotic values of k, it applies for any signal that has $||\vec{X}||_0 \leq k$. Hence, the regime of $\delta = 0$ (esp. constant k) is covered by the FFAST algorithm designed to operate for some $\delta > 0$, at the expense of being sub-optimal in the sample and the computational complexity.

Remark 3.3.2. *[Oversampling ratio r] The minimum achievable oversampling ratio r depends on the number of stages d used in the FFAST architecture. The number of stages d, in turn, is a function of the sparsity index δ (recall $k \propto n^{\delta}$), and increases as $\delta \to 1$ (i.e., as the number of the non-zero coefficients, k, approach the linear regime in n). In Sections 3.5 and 3.6, we show how the number of stages d increase as δ approaches 1. Table 3.2 provides some example values of r and d for different values of the sparsity index δ. In Fig. 3.2 we plot r as a function of δ for a sparsity regime of practical interest, i.e., $0 < \delta < 0.99$.*

δ	1/3	2/3	0.99	0.999	0.9999
d	3	6	8	11	14
r	2.45	3.14	3.74	4.64	5.51

Table 3.2: The table shows the number of subsampling stages d used in the FFAST architecture, and the corresponding values of the oversampling ratio r (for different values of the sparsity index δ).

3.4 DFT using decoding on sparse-graphs

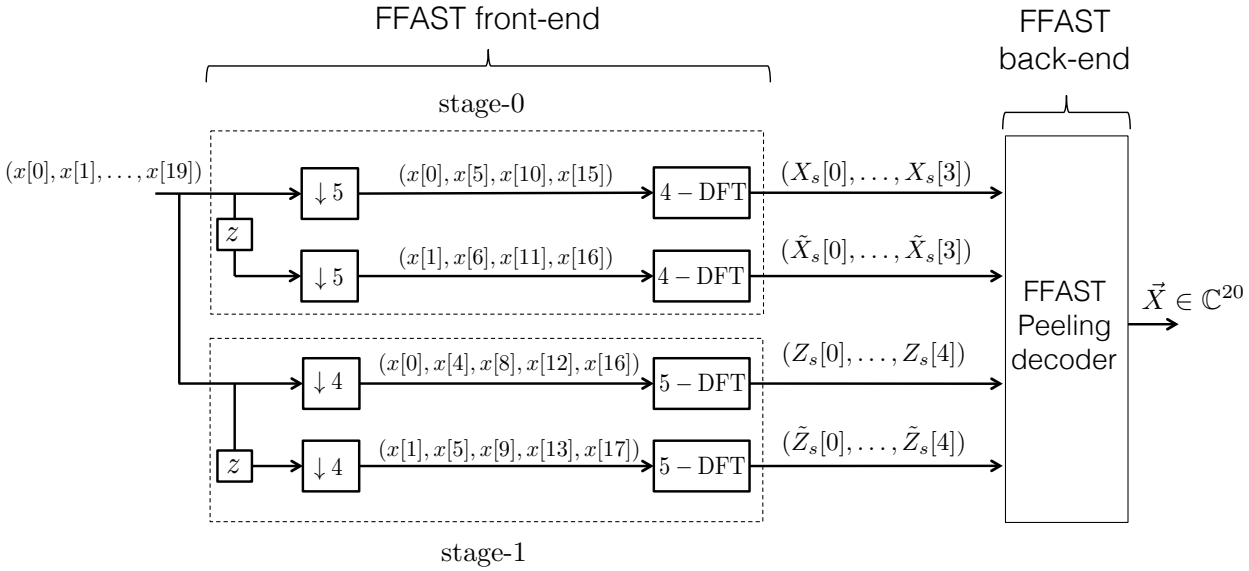


Figure 3.3: A toy-example of the FFAST architecture. The input to the FFAST architecture is a 20-point discrete-time signal $\vec{x} = (x[0], \ldots, x[19])$. The input signal and its circularly shifted version are first subsampled by 5 to obtain two sub-streams, each of length $f_0 = 4$. A 4-point DFT of each sub-stream is then computed to obtain the observations $(X_s[.], \tilde{X}_s[.])$. Similarly, downsampling by 4 followed by a 5-point DFT provides the second set of $f_1 = 5$ observations $(Z_s[.], \tilde{Z}_s[.])$. Note that the number of samples per sub-stream f_0 and f_1 in the two different stages ($d = 2$) are pairwise co-prime and are factors of $n = 20$. In general, the number of stages and the choice of the subsampling factors depend on the sparsity index δ. For the very-sparse regime, $(0 < \delta \leq 1/3)$, the subsampling factors are such that the number of samples per sub-stream in the different stages of the FFAST architecture are relatively co-prime (see Section 3.5 for details). For the less-sparse regime, $(1/3 < \delta < 1)$, the number of samples per sub-stream in different stages have a more complicated "cyclically-shifted" overlapping co-prime structure (see Section 3.6 for details).

In this section, we describe the FFAST sub-sampling "front-end" architecture, as shown in Fig. 3.3, as well as the associated "back-end" FFAST peeling-decoder to compute a k-sparse n-length DFT. Further, we also show an equivalence between computing the sparse DFT of a signal and decoding over an appropriately designed sparse-graph. We use a simple example to illustrate the FFAST sub-sampling front-end and the backend. Consider a 20-point discrete-time signal $\vec{x} = (x[0], \ldots, x[19])$, such that its 20-point DFT $\vec{X}$, is 5-sparse. Let the 5 non-zero DFT coefficients of the signal $\vec{x}$ be $X[1] = 1, X[3] = 4, X[5] = 2, X[10] = 3$

and $X[13] = 7$. The FFAST sub-sampling front-end shown in Fig. 3.3, samples the input signal and its circularly shifted version through multiple stages $d = 2$. Let f_i denote the number of the output samples per delay sub-stream of stage i, e.g., in Fig. 3.3, $f_0 = 4$ and $f_1 = 5$. The FFAST peeling-decoder synthesizes the big DFT $\vec{X}$, from the short DFTs of each of the sub-sampled data streams.

Before we delve into the details of computing the DFT of the signal $\vec{x}$ using the FFAST framework, we review some basic signal processing properties of subsampling-aliasing and circular shifts.

- **Aliasing:** If a signal is subsampled in the time-domain, its frequency components mix together, i.e., alias, in a pattern that depends on the sampling procedure. For example, consider uniform subsampling of $\vec{x}$ by a factor of 5 (see Fig. 3.3) to get $\vec{x}_s = (x[0], x[5], x[10], x[15])$. Then, the 4-point DFT of $\vec{x}_s$ is related to the DFT $\vec{X}$ as:

$$\begin{aligned} X_s[0] &= X[0] + X[4] + X[8] + X[12] + X[16] = 0 \\ X_s[1] &= X[1] + X[5] + X[9] + X[13] + X[17] = 10 \\ X_s[2] &= X[2] + X[6] + X[10] + X[14] + X[18] = 3 \\ X_s[3] &= X[3] + X[7] + X[11] + X[15] + X[19] = 4 \end{aligned}$$

  More generally, if the sampling period is N (we assume that N divides n) then,

$$X_s[i] = \sum_{j \equiv (i)_{n/N}} X[j], \tag{3.4}$$

  where the notation $j \equiv (i)_{n/N}$, denotes $j \equiv i \bmod n/N$.

- **Circular Shift in time:** A circular shift in the time-domain results in a phase shift in the frequency-domain. Consider a circularly shifted signal $\vec{x}^{(1)}$ obtained from $\vec{x}$ as $x[i]^{(1)} = x[(i+1)_n]$. The DFT coefficients of the shifted signal $\vec{x}^{(1)}$, are given as, $X^{(1)}[j] = \omega_n^j X[j]$, where $\omega_n = \exp(2\pi\imath/n)$ is an n^{th} root of unity. In general a circular shift of n_0 results in $X^{(n_0)}[j] = \omega_n^{jn_0} X[j]$.

Using the above signal processing properties of sub-sampling and circular shift, we can compute the relation between the DFT $\vec{X}$ and the output of the FFAST front-end. Next, we group the output of the FFAST front-end into "bin-observation" as follows:

Bin observation

A bin-observation is a 2-dimensional vector formed by collecting one scalar output value from each of the 2 delay chains in a stage. For example, $\vec{y}_{b,0,1}$ is an observation vector of bin 1 in stage 0 and is given by,

$$\vec{y}_{b,0,1} = \begin{pmatrix} X_s[1] \\ \tilde{X}_s[1] \end{pmatrix}. \tag{3.5}$$

The first index of the observation vector corresponds to the stage number, while the second index is the bin number within a stage. Note, that in the FFAST architecture of Fig. 3.3, there are total of 4 bins in stage 0 and 5 bins in stage 1.

Using the above 20-point example signal $\vec{x}$, we explain how to compute the sparse DFT $\vec{X}$, via decoding over an appropriately designed sparse graph.

3.4.1 Computing a sparse DFT is equivalent to decoding on a sparse-graph

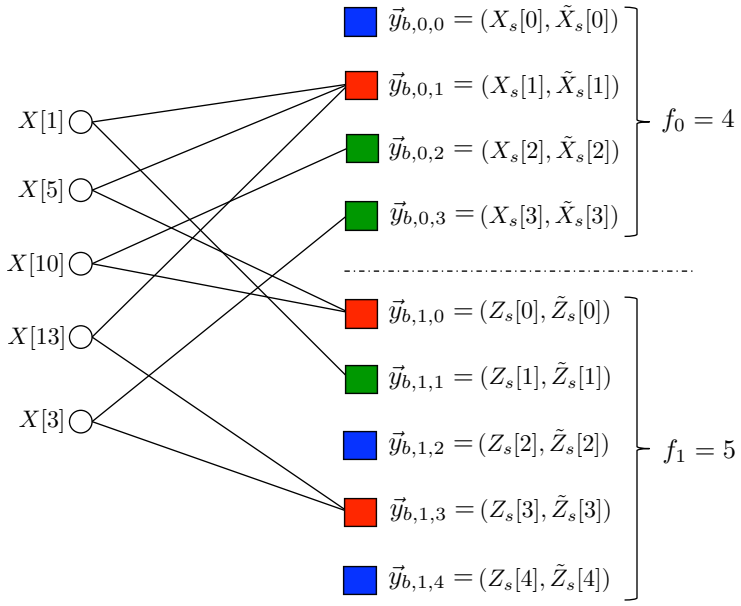


Figure 3.4: A 2-left regular degree bi-partite graph representing the relation between the unknown non-zero DFT coefficients and the observations obtained through the FFAST architecture shown in Fig. 3.3, for the 20-point example signal $\vec{x}$. Variable (left) nodes correspond to the non-zero DFT coefficients and the check (right) nodes are the observations. The observation at each check node is a 2-dimensional complex-valued vector e.g., $\vec{y}_{b,0,0} = (X_s[0], \tilde{X}_s[0])$.

Let, the input signal $\vec{x}$ be processed through the 2 stage FFAST architecture of Fig. 3.3, to obtain the bin-observations $(X_s[\cdot], \tilde{X}_s[\cdot])$ and $(Z_s[\cdot], \tilde{Z}_s[\cdot])$. Then, the relation between the resulting bin-observations and the non-zero DFT coefficients $\vec{X}$ can be computed using the signal processing properties of sub-sampling and circular shift. A graphical representation of this relation is shown in Fig. 3.4. Left nodes of the graph in Fig. 3.4 represent the non-zero DFT coefficients and the right nodes represent the "bins" (check nodes) with vector observations. An edge connects a left node to a right check node iff the corresponding non-zero DFT coefficient contributes to the observation vector of that particular check node, e.g., after aliasing, due to sub-sampling, the DFT coefficient $X[10]$ contributes to the observation vector of bin 2 of stage 0 and bin 0 of stage 1.

We define the following:

- **zero-ton**: A bin that has no contribution from any of the non-zero DFT coefficients of the signal, e.g., bin 0 of stage 0 or bin 2 of stage 1, as shown in Fig. 3.4. A zero-ton bin can be trivially identified from its observations.

- **single-ton**: A bin that has contribution from exactly one non-zero DFT coefficient of the signal, e.g., bin 2 of stage 0. Using the signal processing properties the observation vector of bin 2 of stage 0 is given as,

$$\vec{y}_{b,0,2} = \begin{pmatrix} X[10] \\ e^{2\pi\imath 10/20} X[10] \end{pmatrix}.$$

  The observation vector of a singleton bin can be used to determine the support and the value, of the only non-zero DFT coefficient contributing to that bin, as follows:

  - *support*: The support of the non-zero DFT coefficient contributing to a singleton bin can be computed as,

$$10 = \frac{20}{2\pi} \angle \vec{y}_{b,0,2}[1] y^{\dagger}_{b,0,2}[0] \tag{3.6}$$

  - *Value*: The value of the non-zero DFT coefficient is given by the observation $y_{b,0,2}[0]$.

  We refer to this procedure as a "ratio-test", in the sequel. Thus, a simple ratio-test on the observations of a singleton bin correctly identifies the support and the value of the only non-zero DFT coefficient connected to that bin. It is easy to verify that this property holds for all the singleton bins.

- **multi-ton**: A bin that has a contribution from more than one non-zero DFT coefficients of the signal, e.g., bin 1 of stage 0. The observation vector of bin 1 of stage 0 is,

$$\begin{aligned} \vec{y}_{b,0,1} &= X[1] \begin{pmatrix} 1 \\ e^{\imath 2\pi/20} \end{pmatrix} + X[5] \begin{pmatrix} 1 \\ e^{\imath 2\pi 5/20} \end{pmatrix} + X[13] \begin{pmatrix} 1 \\ e^{\imath 2\pi 13/20} \end{pmatrix} \\ &= \begin{pmatrix} 10 \\ -3.1634 - \imath 3.3541 \end{pmatrix} \end{aligned}$$

  Now, if we perform the "ratio-test" on these observations, we get, the support to be 12.59. Since, we know that the support has to be an integer value between 0 to 19, we conclude that the observations do not correspond to a singleton bin. In Section 2.A, we rigorously show that the ratio-test identifies a multi ton bin almost surely.

Hence, using the "ratio-test" on the bin-observations, the FFAST decoder can determine if a bin is a zero-ton, a single-ton or a multi-ton, almost surely. Also, when a bin is singleton the ratio-test provides the support as well as the value of the non-zero DFT coefficient connected to that bin. We use the following peeling-decoder on the graph in Fig. 3.4, to compute the support and the values of the non-zero DFT coefficients of $\vec{x}$.

3.4.2 FFAST peeling-decoder

Algorithm 3 FFAST Algorithm

Inputs:
- A discrete time signal $\vec{x}$ of length n, whose n-point DFT $\vec{X}$ has at most k non-zero coefficients.
- The subsampling parameters of the FFAST architecture (see Fig. 3.1): 1) number of stages d. and 2) number of samples per sub-stream in each of the d stages $\mathcal{F} = \{f_0, f_1, \ldots, f_{d-1}\}$, chosen as per discussion in Sections 3.5 and 3.6.

Output: An estimate of the k-sparse n-point DFT $\vec{X}$.

FFAST Decoding: Set the initial estimate of the n-point DFT $\vec{X} = 0$. Let ℓ denote the number of iterations performed by the FFAST decoder.

for each iteration **do**
  for each stage $i = 0$ to $d-1$ **do**
    for each bin $j = 0$ to $f_i - 1$ **do**
      if $||\vec{y}_{b,i,j}||^2 == 0$ **then**
        bin j of stage i is a zero-ton.
      else
        (singleton, v_p, p) = ***Singleton-Estimator*** $(\vec{y}_{b,i,j})$.
        if singleton = 'true' **then**
          Peeling: $\vec{y}_{b,s,q} = \vec{y}_{b,s,q} - v_p \begin{pmatrix} 1 \\ e^{\imath 2\pi p/n} \end{pmatrix}$, for all stages s and bins $q \equiv p \bmod f_q$.
          Set, $X[p] = v_p$.
        else
          bin j of stage i is a *multi-ton*.
        end if
      end if
    end for
  end for
end for

The FFAST decoder repeats the following steps (also see pseudocode in Algorithm 3 and Algorithm 4):

1. Select all the edges in the graph with right degree 1 (edges connected to single-tons).
2. Remove these edges from the graph as well as the associated left and right nodes.
3. Remove all the other edges that were connected to the left nodes removed in step-2. When a neighboring edge of any right node is removed, its contribution is subtracted from that check node.

Decoding is successful if, at the end, all the edges have been removed from the graph. It is easy to verify that performing the peeling procedure on the example graph of Fig. 3.4 results in successful decoding, with the coefficients being uncovered in the following possible order: $X[10], X[3], X[1], X[5], X[13]$.

Algorithm 4 Singleton-Estimator

Input: The bin observation $\vec{y}_{b,i,j}$.

Outputs: 1) A boolean flag 'singleton', 2) Estimated value v_p of the non-zero DFT coefficient at position p.

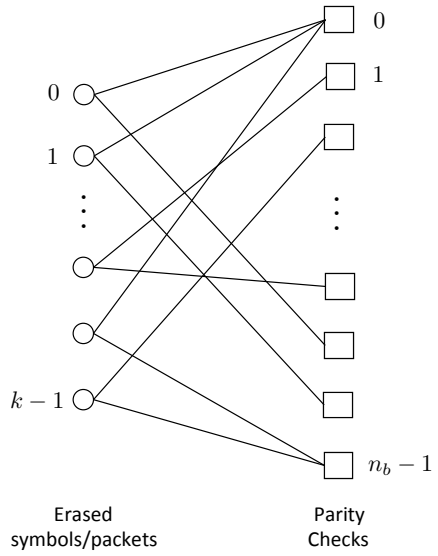
Singleton-Estimator: Set the singleton = 'false'.
if $|y_{b,i,j}[0]| == |y_{b,i,j}[1]|$ and $(n/2\pi)\angle y_{b,i,j}[1]y_{b,i,j}[0]^{\dagger} \in \{0, 1, \ldots, (n-1)\}$ **then**
  singleton = 'true'.
  $v_p = y_{b,i,j}[0]$.
  $p = (n/2\pi)\angle y_{b,i,j}[1]y_{b,i,j}[0]^{\dagger}$.
end if

Thus, the FFAST architecture has *transformed the problem of computing the DFT of $\vec{x}$ into that of decoding over a sparse bi-partite graph* of Fig. 3.4. Clearly the success the FFAST decoder depends on the properties of the sparse bi-partite graph resulting from the sub-sampling operation of FFAST front-end.

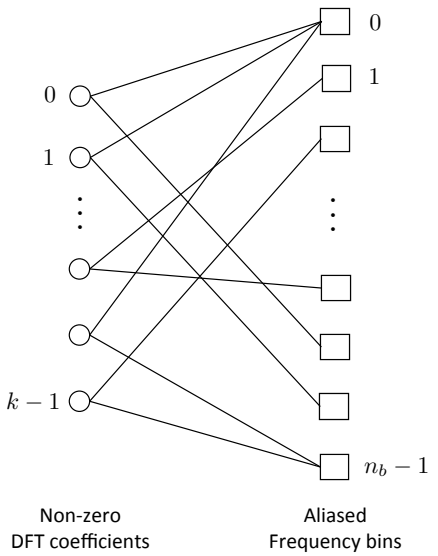
3.4.3 Connection to coding for packet erasure channels

The problem of decoding over sparse bi-partite graphs has been well studied in the coding theory literature. In this section we draw an analogy between decoding over sparse-graph codes for a packet erasure channel and decoding over bi-partite graphs induced by the FFAST architecture.

Consider an $(n, n-n_b)$ packet erasure code. Each n-length codeword consists of $(n-n_b)$ information packets and n_b parity packets. Suppose that the code is used over an erasure



(a) Bi-partite graph, after subtracting the contribution of the correctly received $n-k$ packets, corresponding to a parity check matrix of a sparse-graph code designed for an erasure channel.



(b) Bi-partite graph representing the connections (aliasing) among the non-zero DFT coefficients due to subsampling, and the observations obtained by the FFAST decoder.

Figure 3.5: Comparison between the bi-partite graphs corresponding to the parity check matrix of a sparse-graph code for an erasure channel and a graph induced by the FFAST architecture.

channel that uniformly at random drops some k number of packets. A bi-partite graph representation of the parity check matrix of the code, after subtracting the contribution of the correctly received packets, is shown in Fig. 3.5(a). In Table 3.3 we provide comparison between decoding over bi-partite graphs of Fig. 3.5(a) and Fig. 3.5(b).

Thus, the problem of decoding over bi-partite graphs induced by the FFAST architecture is closely related to the decoding of sparse-graph codes for an erasure-channel. We use this analogy: a) to design a set of uniform sub-sampling patterns that induce 'good' left-regular degree sparse-graph codes; and b) to formally connect our proposed Chinese-Remainder-Theorem based aliasing framework to random sparse-graph codes constructed using a balls-and-bins model, and analyze the convergence behavior of our algorithm using density evolution techniques.

Next, we address the question of how to carefully design the sub-sampling parameters of the FFAST front-end architecture so as to get "good-graphs".

Erasure Channel	Sparse DFT
1. Explicitly designed sparse-graph code.	1. Implicitly designed sparse-graph code induced by sub-sampling.
2. $n-k$ correctly received packets.	2. $n-k$ zero DFT coefficients.
3. k-erased packets.	3. k unknown non-zero DFT coefficients
4. Peeling-decoder recovers the values of the erased packets using 'singleton' check nodes. The locations of which packets are erased are *known*.	4. Peeling-decoder recovers *both* the values and the locations of the non-zero DFT coefficients using 'singleton' check nodes. The locations of the non-zero DFT coefficients are *not known*. This results in a $2\times$ cost in the sample complexity.
5. Codes based on regular-degree bipartite graphs are near-capacity-achieving. More efficient, capacity-achieving irregular-degree bipartite graph codes can be designed.	5. The FFAST architecture based on uniform subsampling can induce only left-regular degree bi-partite graphs.

Table 3.3: Comparison between decoding over a sparse-graph code for a packet erasure channel and computing a sparse DFT using the FFAST architecture.

3.5 Performance analysis of the FFAST algorithm for the *very-sparse* ($k \propto n^{\delta},\ 0 < \delta \leq 1/3$) regime

In the previous section, we showed that the problem of computing a k-sparse n-point DFT of a signal can be transformed into a problem of decoding over sparse bipartite graphs using the FFAST architecture. In this section, we describe how to choose a set of uniform sub-sampling patterns, guided by the CRT, to induce a good sparse-graph code. As shown in Section 3.4.3, the FFAST decoding process is closely related to the decoding procedure on sparse-graph codes designed for erasure channels. From the coding theory literature, we know that there exist several sparse-graph code constructions that are low-complexity and capacity-achieving for the erasure channels. The catch for us is that we are not at liberty to use any arbitrary bi-partite graph, but *can choose only those graphs that can be induced through our proposed subsampling*. Next, we show that a *deterministic* bi-partite graph construction based on the Chinese-Remainder-Theorem (CRT), in conjunction with a *uniformly random* support of the non-zero DFT coefficients, creates sparse-graph codes that: a) have all the good properties that are sufficient for reliable decoding; and b) can be induced using the FFAST subsampling architecture.

We describe two ensembles of bi-partite graphs, the first based on a "balls-and-bins"

model, and the second using a deterministic construction based on the CRT. Later, in Lemma 3.5.1, we show that the two ensembles are equivalent.

Let $\mathcal{F} = \{f_0, \ldots, f_{d-1}\}$, be a set of pairwise co-prime integers and $n_b \triangleq \sum_{i=0}^{d-1} f_i$. As explained in Table 3.1, the integers f_i's are the number of samples per sub-stream in the d stages of the FFAST architecture (see Fig. 3.1). The integers f_i's are approximately equal and we use $\mathbf{F}$ to denote this value. More precisely, $f_i = \mathbf{F} + O(1)$ for $i = 0, \ldots, d-1$, where $\mathbf{F}$ is an asymptotically large number. The $O(1)$ perturbation term in each f_i is used to obtain a set of co-prime integers[4] approximately equal to $\mathbf{F}$. Note, that the total number of samples used by the FFAST algorithm is $m = 2d\mathbf{F} + O(1)$ (see Fig. 3.1). In this section, we use $\mathbf{F} = \eta k$, for some constant η, which results in an order optimal sample complexity $m = O(k)$, but the constructions and results trivially extend to the case when $m > \Omega(k)$.

3.5.1 Randomized construction based on the "Balls-and-Bins" model: $\mathcal{C}_1^k(\mathcal{F}, n_b)$

We construct a bi-partite graph with k *variable nodes* on the left and n_b *check nodes* on the right. The bipartite graphs that we consider in this chapter have each variable node v connected to exactly d check nodes, while the edge degree of the check nodes is variable, i.e., left-regular degree bi-partite graphs. An example graph from an ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, for $\mathcal{F} = \{4, 5\}, d = 2, k = 5$ and $n_b = 9$ is provided in Fig. 3.4. More generally, the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ of d-left regular degree bipartite graphs constructed using a "balls-and-bins" model is defined as follows. Set $n_b = \sum_{i=0}^{d-1} f_i$, where $\mathcal{F} = \{f_i\}_{i=0}^{d-1}$. Partition the set of n_b check nodes into d subsets with the i^{th} subset having f_i check nodes. For each variable node, choose one neighboring check node in each of the d subsets, uniformly at random. The corresponding d-left regular degree bipartite graph is then defined by connecting the variable nodes with their neighboring check nodes by an undirected edge.

An *edge* e in the graph is represented as a pair of nodes $e = \{v, c\}$, where v and c are the variable and check nodes incident on the edge e. By a *directed* edge $\vec{e}$ we mean an ordered pair (v, c) or (c, v) corresponding to the edge $e = \{v, c\}$. A *path* in the graph is a directed sequence of directed edges $\vec{e_1}, \ldots, \vec{e_t}$ such that, if $\vec{e_i} = (u_i, u_i')$, then the $u_i' = u_{i+1}$ for $i = 1, \ldots, t-1$. The length of the path is the number of directed edges in it, and we say that the path connecting u_1 to u_t starts from u_1 and ends at u_t.

Directed Neighborhood

The *directed neighborhood of depth* ℓ of $\vec{e} = (v, c)$, denoted by $\mathcal{N}_{\vec{e}}^{\ell}$, is defined as the induced subgraph containing all the edges and nodes on paths $\vec{e_1}, \ldots, \vec{e_\ell}$ starting at node v such that $\vec{e_1} \neq \vec{e}$. An example of a directed neighborhood of depth $\ell = 2$ is given in Fig. 3.6. If the

[4] An example construction of an approximately equal sized 3 co-prime integers can be obtained as follows. Let $\mathbf{F} = 2^{r_0} 3^{r_1} 5^{r_2}$ for any integers r_0, r_1, r_2 greater than 1. Then, $f_0 = \mathbf{F} + 2, f_1 = \mathbf{F} + 3$ and $f_2 = \mathbf{F} + 5$ are co-prime integers.

induced sub-graph corresponding to the directed neighborhood $\mathcal{N}_{\vec{e}}^{\ell}$ is a tree then we say that the neighborhood of the edge $\vec{e}$ is *tree-like*.

3.5.2 Ensemble of bipartite graphs constructed using the Chinese-Remainder-Theorem (CRT): $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$

First set $n = \prod_{i=0}^{d-1} f_i$ and $n_b = \sum_{i=0}^{d-1} f_i$, where $\mathcal{F} = \{f_i\}_{i=0}^{d-1}$. The ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ of d-left regular degree bipartite graphs, with k variable nodes and n_b check nodes, is defined as follows. Partition the set of n_b check nodes into d subsets with the i^{th} subset having f_i check nodes (see Fig. 3.4 for an example). Consider a set $\mathcal{I}$ of k integers, where each element of the set $\mathcal{I}$ is chosen uniformly at random, with replacement, between 0 and $n-1$. Assign the k integers from the set $\mathcal{I}$ to the k variable nodes in an arbitrary order. Label the check nodes in the set i from 0 to $f_i - 1$ for all $i = 0, \ldots, d-1$. A d-left regular degree bi-partite graph with k variable nodes and n_b check nodes, is then obtained by connecting a variable node with an associated integer v to a check node $(v)_{f_i}$ in the set i, for $i = 0, \ldots, d-1$. The ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ is a collection of all the d-left regular degree bipartite graphs induced by all possible sets $\mathcal{I}$. A uniformly random choice of integers in the set $\mathcal{I}$, implies that all the graphs in the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ occur with equal probability.

Note that the modulo rule used to generate a graph in the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ is same as the one used in equation (3.4) of Section 3.4. Thus, the FFAST architecture of Fig. 3.3 described in Section 3.4, *generates graphs from the CRT ensemble* $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$, where the indices $\mathcal{I}$ of the k variable nodes are the locations (or support) of the non-zero DFT coefficients[5] of the signal $\vec{x}$.

Lemma 3.5.1. *The ensemble of bipartite graphs* $\mathcal{C}_1^k(\mathcal{F}, n_b)$ *is identical to the ensemble* $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$.

Proof. It is trivial to see that $\mathcal{C}_2^k(\mathcal{F}, n, n_b) \subset \mathcal{C}_1^k(\mathcal{F}, n_b)$. Next we show the reverse. Consider a graph $\mathcal{G}_1 \in \mathcal{C}_1^k(\mathcal{F}, n_b)$. Suppose, a variable node $v \in \mathcal{G}_1$ is connected to the check nodes numbered $\{r_i\}_{i=0}^{k-1}$. Then, using the CRT, one can find a unique integer q between 0 and $n-1$ such that $(q)_{f_i} = r_i \ \forall i = 0, \ldots, d-1$. Thus, for every graph $\mathcal{G}_1 \in \mathcal{C}_1^k(\mathcal{F}, n_b)$, there exists a set $\mathcal{I}$ of k integers, that will result in an identical graph using the CRT based construction. Hence, $\mathcal{C}_1^k(\mathcal{F}, n_b) = \mathcal{C}_2^k(\mathcal{F}, n, n_b)$. □

Next, we analyze the performance of the peeling-decoder over a random choice of a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, which, using the Lemma 3.5.1, along with the fact that all the graphs in both the ensembles occur with equal probability, provides a lower bound on the probability of successful decoding of the FFAST decoder over graphs in the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$.

[5]The integers in the set $\mathcal{I}$ are chosen uniformly at random, with replacement, between 0 and $n-1$. A set $\mathcal{I}$ with repeated elements then corresponds to a signal with fewer than k non-zero DFT coefficients.

3.5.3 Performance analysis of the peeling-decoder on graphs from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$

In this section we analyze the probability of success of the peeling-decoder, over a randomly chosen graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, after a fixed number of iterations ℓ. Our analysis follows exactly the arguments in [54] and [67]. Thus, one may be tempted to take the results from [54] "off-the-shelf". However, we choose here to provide a detailed analysis for two reasons. First, our graph construction in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ is different from that used in [54], which results in some fairly important differences in the analysis, such as the expansion properties of the graphs, thus warranting an independent analysis. Secondly we want to make this thesis more self-contained and complete.

We now provide a brief outline of the proof elements highlighting the main technical components needed to show that the peeling-decoder successfully decodes all the non-zero DFT coefficients with high probability.

- *Density evolution:* We analyze the performance of the message-passing algorithm, over a typical graph from the ensemble, for ℓ iterations. First, we assume that a local neighborhood of depth 2ℓ of every edge in a typical graph in the ensemble is tree-like, i.e., cycle-free. Under this assumption, all the messages between variable nodes and the check nodes, in the first ℓ rounds of the algorithm, are independent. Using this independence assumption, we derive a recursive equation that represents the expected evolution of the number of singletons uncovered at each round for this typical graph.

- *Convergence to the cycle-free, case:* Using a Doob martingale as in [54], we show that a random graph from the ensemble, chosen as per nature's choice of the non-zero DFT coefficients, behaves like a "typical" graph, i.e., 2ℓ-depth neighborhood of most of the edges in the graph is cycle-free, with high probability. This proves that for a random graph in $\mathcal{C}_1^k(\mathcal{F}, n_b)$, the peeling-decoder decodes all but an arbitrarily small fraction of the variable nodes with high probability in a constant number of iterations, ℓ.

- *Completing the decoding using the graph expansion property:* We first show that if a graph is an "expander" (as will be defined later in Section 3.5.3), and the peeling-decoder successfully decodes all but a small fraction of the non-zero DFT coefficients, then it decodes all the non-zero DFT coefficients successfully. Next, we show that a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ is an expander with high probability.

Density evolution for local tree-like view

In this section we assume that a local neighborhood of depth 2ℓ of every edge in a typical graph from the ensemble is tree-like. Next, we define the edge-degree distribution polynomials of the bipartite graphs as $\lambda(\alpha) \triangleq \sum_{i=1}^{\infty} \lambda_i \alpha^{i-1}$ and $\rho(\alpha) \triangleq \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}$, where λ_i (resp. ρ_i) denotes the probability that an edge of the graph is connected to a left (resp. right)

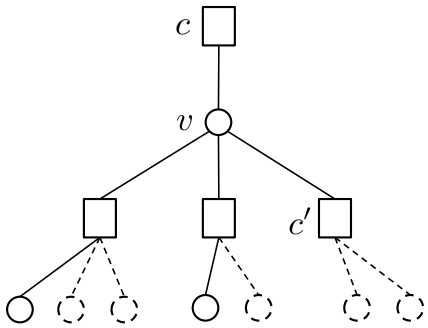


Figure 3.6: Directed neighborhood of depth 2 of an edge $\vec{e} = (v, c)$. The dashed lines correspond to nodes/edges removed at the end of iteration j. The edge between v and c can be potentially removed at iteration $j+1$ as one of the check nodes c' is a singleton (it has no more variable nodes remaining at the end of iteration j).

node of degree i. Thus for the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, constructed using the balls-and-bins procedure, $\lambda(\alpha) = \alpha^{d-1}$ by construction. Further, as shown in Appendix 3.A, the edge degree distribution $\rho(\alpha) = \exp(-(1-\alpha)/\eta)$.

Let p_j denote the probability that an edge is present after round j of the peeling-decoder, then $p_0 = 1$. Under the tree-like assumption, the probability p_{j+1}, is given as,

$$p_{j+1} = \lambda(1 - \rho(1 - p_j)) \quad j = 0, 1, \ldots, \ell - 1. \tag{3.7}$$

Equation (3.7) can be understood as follows (also see Fig. 3.6): the tree-like assumption implies that, up to iteration ℓ, messages on different edges are independent. Thus, the total probability, that at iteration $j+1$, a variable node v is *decoded* due to a particular check node is given by $\rho(1-p_j) = \sum_{i=1}^{\infty} \rho_i (1-p_j)^{i-1}$ and similarly the total probability that none of the neighboring check nodes decode the variable node v is $p_{j+1} = \lambda(1 - \rho(1 - p_i))$. Specializing equation (3.7) for the edge degree distributions of $\mathcal{C}_1^k(\mathcal{F}, n_b)$ we get,

$$p_{j+1} = \left(1 - e^{-\frac{p_j}{\eta}}\right)^{d-1}, \ \forall\ j = 0, 1, \ldots, \ell - 1 \tag{3.8}$$

where $p_0 = 1$. The evolution process of (3.8) asymptotically (in the number of iterations ℓ) converges to 0 for appropriate choice of the parameter η, e.g., see Table 3.4.

d	2	3	4	5	6	7	8	9
η	1.0000	0.4073	0.3237	0.2850	0.2616	0.2456	0.2336	0.2244
$d\eta$	2.0000	1.2219	1.2948	1.4250	1.5696	1.7192	1.8688	2.0196

Table 3.4: Minimum value of η, the average number of bins or check nodes per stage per variable node, required for the density evolution of (3.8) to converge asymptotically. The threshold η depends on the number of stages d. Although, the minimum threshold of η decreases with increasing d, the total number ($\approx d\eta k$) of bins or check nodes required for the convergence of (3.8) is a not a monotonic function in d.

Convergence to cycle-free case

In Lemma 3.5.2 we show that: a) the expected behavior over all the graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ converges to that of a cycle-free case, and b) with exponentially high probability, almost all the edges in a random graph in the ensemble have a tree-like neighborhood and the proportion of the edges that are not decoded after ℓ iterations of the peeling-decoder is tightly concentrated around p_ℓ, as defined in (3.8).

Lemma 3.5.2 (Convergence to Cycle-free case)**.** *Over the probability space of all graphs $\mathcal{C}_1^k(\mathcal{F}, n_b)$, let Z be the total number of edges that are not decoded after ℓ (an arbitrarily large but fixed) iterations of the peeling-decoder over a randomly chosen graph. Further let p_ℓ be as given in the recursion (3.8). Then there exist constants β and γ such that for any $\epsilon_1 > 0$ and sufficiently large k we have*

$$(a) \qquad \mathbb{E}[Z] < kdp_\ell + \frac{d\gamma}{\eta}; \tag{3.9}$$

$$(b) \qquad Pr\left(|Z - kdp_\ell| > kd\epsilon\right) < 2e^{-\beta\epsilon_1^2 k}, \tag{3.10}$$

where $\gamma > 0$ is a constant.

Proof. Please see Appendix 3.D. □

Successful Decoding using Expansion

In the previous section we showed that with high probability, i.e., with probability approaching 1 exponentially in k, the peeling-decoder decodes all but an arbitrarily small fraction of variable nodes. In this section, we show how to complete the decoding if the graph is a "good-expander". Our problem requires the following definition of an "expander-graph", which is somewhat different from conventional notions of an expander-graph in literature, e.g., *edge expander, vertex expander or spectral expander* graphs.

Definition 3.5.3 (Expander graph)**.** *A d-left regular degree bipartite graph constructed as described in Section 3.5.1 is called an (α, β, d) expander, if for all subsets S, of variable nodes, of size at most αk, there exists a right neighborhood of S, i.e., $N_i(S)$, that satisfies $|N_i(S)| > \beta|S|$ for some $i = 0, \ldots, d-1$.*

In the following lemma, we show that if a graph is an expander, and if the peeling-decoder successfully decodes all but a small fraction of the non-zero DFT coefficients, then it decodes all the non-zero DFT coefficients successfully.

Lemma 3.5.4. *Consider a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, with $|\mathcal{F}| = d$, that is an $(\alpha, 1/2, d)$ expander for some $\alpha > 0$. If the peeling-decoder over this graph succeeds in decoding all but at most αk variable nodes, then it decodes all the variable nodes.*

Proof. See Appendix 3.B □

In Lemma 3.5.5, we show that most of the graphs in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ are expanders.

Lemma 3.5.5. *Consider a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $d \geq 3$. Then, all the subsets S of the variable nodes, of the graph, satisfy $\max\{|N_i(S)|\}_{i=0}^{d-1} > |S|/2$,*

a) with probability at least $1 - e^{-\epsilon k \log(n_b/k)}$, for sets of size $|S| = \alpha k$, for small enough $\alpha > 0$ and some $\epsilon > 0$.

b) with probability at least $1 - O(1/n_b)$, for sets of size $|S| = o(k)$.

Proof. See Appendix 3.C □

The condition $d \geq 3$ is a necessary condition for part (b) of Lemma 3.5.5. This can be seen as follows. Consider a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $|\mathcal{F}| = d$. If any two variable nodes in the graph have the same set of d neighboring check nodes, then the expander condition, for the set S consisting of these two variable nodes, will not be satisfied. In a bi-partite graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, there are a total of $O(\mathbf{F}^d)$ distinct sets of d check nodes. Each of the k variable nodes chooses a set of d check nodes, uniformly at random and with replacement, from the total of $O(\mathbf{F}^d)$ sets. If we assign p people a uniformly at random birthday between 0 to $N-1$, the probability $Pr(p; N)$ that at least two people have same birthday is given by,

$$Pr(p; N) \approx 1 - e^{-p^2/2N}. \tag{3.11}$$

This is also known as the *birthday paradox* or the *birthday problem* in literature [59]. For a graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, we have $N = O(\mathbf{F}^d)$ and $p = k$, where $\mathbf{F} = \eta k$ for some constant $\eta > 0$. Hence, if the number of stages $d \leq 2$, there is a constant probability of *birthday clash*, i.e., there exists a pair of variable nodes that share the same neighboring check nodes, in both stages, thus violating the expander condition.

Theorem 3.5.6. *The peeling-decoder over a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $d \geq 3$ and $\mathbf{F} = \eta k$:*

a) successfully uncovers all the variable nodes with probability at least $1 - O(1/n_b)$;

b) successfully uncovers all but a vanishingly small fraction, i.e., $o(k)$, of the variable nodes with probability at least $1 - e^{-\epsilon k \log(n_b/k)}$, for some $\epsilon > 0$,

for an appropriate choice of the constant η as per Table 3.4.

Proof. Let Z be the number of edges not decoded by the peeling-decoder in ℓ (large but fixed constant) iterations. Then, from recursion (3.8) and Lemma 3.5.2, for an appropriate choice of the constant η (as per Table 3.4), $Z \leq \alpha k$, for an arbitrarily small constant $\alpha > 0$, with probability at least $1 - e^{-\beta \epsilon_1^2 k}$. The result then follows from Lemmas 3.5.5 and 3.5.4. □

3.5.4 Performance of the FFAST-decoder over graphs in $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ for $k \propto n^\delta$, for $(0 < \delta \leq 1/3)$.

In Section 3.5.3 we analyzed the performance of a simple iterative peeling decoder over graphs chosen randomly from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, constructed using a balls-and-bins model. In this section we connect this discussion to the problem of computing a k-sparse n-point DFT of a signal, where $k \propto n^\delta$, and $(0 < \delta \leq 1/3)$.

Consider an ensemble $\mathcal{C}_2^k(\mathcal{F}, \tilde{n}, n_b)$ generated by a $d = 3$ stage FFAST architecture. Let $\mathcal{F} = \{f_0, f_1, f_2\}$ be a set of co-prime integers such that each $f_i = \mathbf{F} + O(1)$ and $\mathbf{F} = \eta k$, where η is chosen as per Table 3.4 for $d = 3$. Also, note that $\tilde{n} = \prod_{i=0}^{2} f_i$ and $n_b = \sum_{i=0}^{2} f_i$.

Consider a signal $\vec{x}$ of length $n = \tilde{n}\mathcal{P}$, for some integer $\mathcal{P} \geq 1$, that has k-sparse DFT $\vec{X}$. For $\mathcal{P} = 1$ the sparsity $k = O(n^{1/3})$ and as $\mathcal{P}$ increases the sparsity index δ approaches 0, i.e., $k \propto n^\delta$, and $0 < \delta \leq 1/3$.

For $\mathcal{P} = 1$, the CRT guarantees that every integer j between 0 and $n-1$ is uniquely represented by a triplet $((j)_{f_0}, (j)_{f_1}, (j)_{f_2})$. However, for $\mathcal{P} > 1$, this is not true. For example, when $\mathcal{P} = 2$, for every triplet (r_0, r_1, r_2) there are exactly two numbers between 0 and $n-1$ that have (r_0, r_1, r_2) as remainders modulo $f_i, i = 0, 1, 2$. Hence, the only difference between $\mathcal{P} = 1$ and $\mathcal{P} > 1$ in terms of the resulting bi-partite graph is whether or not multiple variable nodes (the non-zero DFT coefficients) have identical neighboring check nodes in all the 3 stages. Recall, in the balls-and-bins ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$ described in Section 3.5.1, we did not constrain multiple variable nodes to have distinct set of check nodes (although it was a high probability event when neighbors are chosen uniformly at random). Despite this, we showed in Theorem 3.5.6 that the peeling-decoder successfully decodes the values of all variable nodes with high probability. As a result, the proof of Theorem 3.3.1 for the *very-sparse* regime follows from Lemma 3.5.1 and Theorem 3.5.6. ■

3.6 Performance analysis of the FFAST algorithm for the *less-sparse regime* $(k \propto n^\delta,\ 1/3 < \delta < 1)$

In the previous section, we analyzed the performance of the FFAST algorithm for the very-sparse regime $(k \propto n^\delta,\ 0 < \delta \leq 1/3)$. Recall that for the very-sparse regime, the integers in the set $\mathbf{F} = \{f_0, \ldots, f_{d-1}\}$, i.e., the number of samples per sub-stream in the d different stages, were pairwise co-prime. For the less-sparse regime $(k \propto n^\delta,\ 1/3 < \delta < 1)$ the relation between the integers f_i's is bit more involved. So, for ease of exposition in this section, we describe the achievable FFAST construction for a special case of $k \propto n^{2/3}$, i.e., $\delta = 2/3$, which generalizes to the regimes $\delta = 1 - 1/d$, for integer values of $d \geq 3$, through an induction argument. Later in Section 3.6.3 we show how to achieve the intermediate values of δ. To summarize, our approach for the less-sparse regime is as follows.

- First we analyze the performance of the FFAST algorithm for $\delta = 2/3$. Then, in

Section 3.6.2, we provide a brief sketch of how to generalize the FFAST architecture to any $\delta = 1 - 1/d$, for integer values of $d \geq 3$. This covers the range of values of $\delta = 2/3, 3/4, \ldots$.

- In Section 3.6.3, we show how to achieve the intermediate values of δ, thus covering the entire range of the sparsity index $1/3 < \delta < 1$.

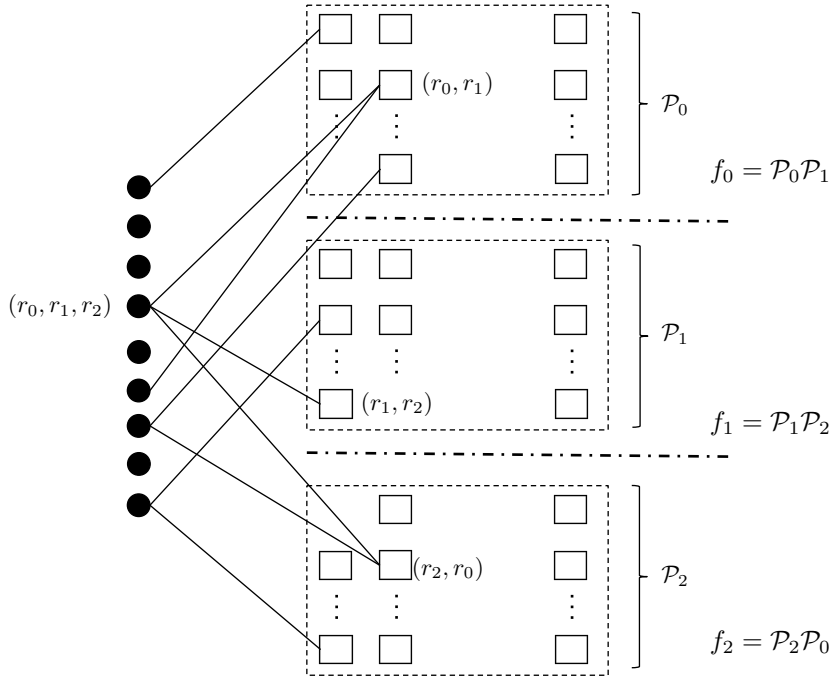


Figure 3.7: A bi-partite graph with k variable nodes and $n_b = \sum_{i=0}^{2} f_i$ check nodes, constructed using a balls-and-bins model. The check nodes in each of the 3 sets are arranged in a matrix format, e.g., the f_0 check nodes in the set 0 are arranged in $\mathcal{P}_0$ rows and $\mathcal{P}_1$ columns. A check node in a row r_0 and column r_1 in the set 0, is indexed by a pair (r_0, r_1) and so on and so forth. Each variable node chooses a triplet (r_0, r_1, r_2), where r_i is between 0 and $\mathcal{P}_i - 1$ uniformly at random. A 3-regular degree bi-partite graph is then constructed by connecting a variable node with a triplet (r_0, r_1, r_2) to the check nodes $(r_0, r_1), (r_1, r_2)$ and (r_2, r_0) in the three sets of the check nodes respectively.

3.6.1 Less-sparse regime of $\delta = 2/3$

Let $\{\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2\}$ be a set of pairwise co-prime integers such that $\mathcal{P}_i = \mathbf{F} + O(1),\ i = 0, 1, 2$, for some large integer $\mathbf{F}$ chosen as follows. Let $n = \prod_{i=0}^{2} \mathcal{P}_i$ and $k \propto \mathbf{F}^2$, i.e., $k \propto n^{2/3}$. Now, set $f_0 = \mathcal{P}_0\mathcal{P}_1,\ f_1 = \mathcal{P}_1\mathcal{P}_2$ and $f_2 = \mathcal{P}_2\mathcal{P}_3$.

Balls-and-Bins construction

We construct a bi-partite graph with k variable nodes on the left and $n_b = \sum_{i=0}^{2} f_i$, check nodes on the right (see Fig. 3.7) using balls-and-bins model as follows. Partition the n_b

check nodes into 3 sets/stages containing f_0, f_1 and f_2 check nodes respectively. The check nodes in each of the 3 sets are arranged in a matrix format as shown in Fig. 3.7, e.g., f_0 check nodes in the set 0 are arranged in $\mathcal{P}_0$ rows and $\mathcal{P}_1$ columns. A check node in row r_0 and column r_1 in the set 0, is indexed by a pair (r_0, r_1) and so on and so forth for all the other check nodes. Each variable node chooses a triplet (r_0, r_1, r_2), where r_i is between 0 and $\mathcal{P}_i - 1$ uniformly at random. The triplets are chosen with replacement and independently across all k variable nodes. A 3-regular degree bi-partite graph with k variable nodes and n_b check nodes is then constructed by connecting a variable node with a triplet (r_0, r_1, r_2) to the check nodes $(r_0, r_1), (r_1, r_2)$ and (r_2, r_0) in the three sets of check nodes respectively, e.g., see Fig. 3.7.

Connection to the CRT based bi-partite graphs induced by the FFAST architecture

Each variable node is associated with an integer v between 0 and $n-1$ (location of the DFT coefficient). As a result of the subsampling and computing a smaller DFTs in the FFAST architecture (see Fig 3.1), a variable node with an index v is connected to the check nodes $(v)_{f_0}, (v)_{f_1}$ and $(v)_{f_2}$ in the 3 stages, in the resulting aliased bi-partite graph. The CRT implies that v is uniquely represented by a triplet (r_0, r_1, r_2), where $r_i = (v)_{\mathcal{P}_i}$. Also, $((v)_{f_i})_{\mathcal{P}_i} = (v)_{\mathcal{P}_i} = r_i$, for all $i = 0, 1, 2$. Thus, the FFAST architecture induces a 3-regular degree bi-partite graph with k variable nodes and n_b check nodes, where a variable node with an associated triplet (r_0, r_1, r_2) is connected to the check nodes $(r_0, r_1), (r_1, r_2)$ and (r_2, r_0) in the three sets respectively. Further, a uniformly random model for the support v of a non-zero DFT coefficient, corresponds to choosing the triplet (r_0, r_1, r_2) uniformly at random. Thus, the CRT based construction, induced by the FFAST architecture, is equivalent to the balls-and-bins construction discussed in the Section 3.6.1.

Following the outline of the proof of Theorem 3.3.1 (provided in Section 3.5), one can show the following, for the balls-and-bins construction of Section 3.6.1:

1. *Density evolution for the cycle-free case*: Assuming a local tree-like neighborhood derive a recursive equation (similar to equation 3.8) representing the expected evolution of the number of singletons uncovered at each round for a "typical" graph from the ensemble.

2. *Convergence to the cycle-free case*: Using a Doob martingale show an equivalent of Lemma 3.5.2 for the less-sparse regime, where the number of check nodes in the 3 different stages f_0, f_1 and f_2 are not pairwise co-prime.

3. *Completing the decoding using the graph expansion property*: A random graph from the ensemble is a good expander with high probability. Hence, if the FFAST decoder successfully decodes all but a constant fraction of variable nodes, it decodes all the variable nodes.

The analysis of the first two items for the less-sparse regime is similar in spirit to the one in Section 3.5, and will be skipped here. However, the analysis of the third item will be described here as there are some key differences, mainly arising from the nature of the overlapping co-prime number of check nodes in the bi-partite graphs for the less-sparse regime in contrast to the very-sparse regime. In Section 3.5, for the very-sparse regime we showed (in Lemma 3.5.5) that the bottleneck failure event is not being able to decode *all* the DFT coefficients. In this section, we analyze this bottleneck failure event for the case of the less-sparse regime. In particular, we show that if the FFAST decoder has successfully decoded all but a small constant number of DFT coefficients, then it decodes all the DFT coefficients successfully with high probability.

Decoding all the variable nodes using the expansion properties of the CRT construction

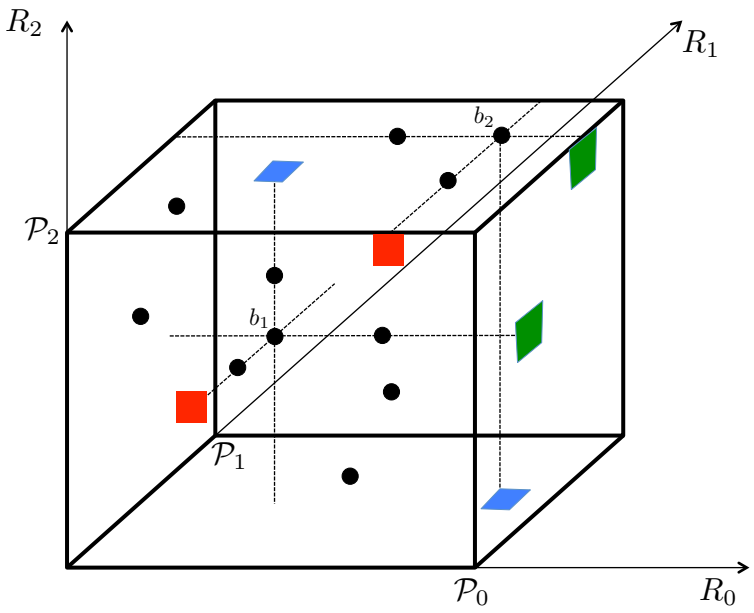


Figure 3.8: A 3D visualization of a bipartite graph from the ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$ corresponding to the less-sparse regime of $\delta = 2/3$. Consider a $3D$ cartesian coordinate space with axes as R_0, R_1, R_2. Recall that for $\delta = 2/3$, $\{\mathcal{P}_j\}_{j=0}^2$ are pairwise co-prime integers, such that $\mathcal{P}_i \approx \mathbf{F}$, $i = 0, 1, 2$, where $\mathbf{F}$ is a large integer. The set $\mathbf{F} = \{f_0, f_1, f_2\}$, where $f_0 = \mathcal{P}_0\mathcal{P}_1$, $f_1 = \mathcal{P}_1\mathcal{P}_2$ and $f_2 = \mathcal{P}_2\mathcal{P}_0$. The parameters $k \propto \mathbf{F}^2$ and $n = \prod_{i=0}^2 \mathcal{P}_i$, i.e., $k \propto n^{2/3}$. A variable node with an associated triplet (r_0, r_1, r_2) is represented by a 'ball' at the position (r_0, r_1, r_2). The f_0 check nodes in stage 1 of the bi-partite graph are represented by 'blue' squares and likewise the ones in f_1 are 'green' and the check nodes in stage f_2 are 'red'. All the neighboring check nodes of a variable node, e.g., b_1, are multi-ton iff there is at least one more variable node along each of the three directions R_0, R_1 and R_2. The green and red neighboring check nodes connected to the ball b_2 are multi-tons, while the blue neighboring check node is a singleton since there are no other variable nodes along the R_2 direction of b_2.

Consider an alternative $3D$ cube visualization of a random bi-partite graph from the

ensemble $\mathcal{C}_2^k(\mathcal{F}, n, n_b)$, for the less-sparse regime, as shown in Fig. 3.8. Consider a $3D$ cartesian coordinate space with axes as R_0, R_1, R_2. A variable node associated with a triplet (r_0, r_1, r_2) is represented by a ball at the position (r_0, r_1, r_2). The plane R_0-R_1 corresponds to the check nodes in stage 0, in a sense that all the variable nodes that have identical (r_0, r_1) but distinct r_2 are connected to the check node (r_0, r_1) and so on. Similarly the planes R_1-R_2 and R_2-R_0 correspond to the check nodes in stages 1 and 2 respectively. Thus, a variable node with co-ordinates (r_0, r_1, r_2) is connected to a multi-ton check nodes in all the 3 stages, if and only if there exist variable nodes with co-ordinates (r_0, r_1, r_2'), (r_0', r_1, r_2) and (r_0, r_1', r_2) (see Fig. 3.8), i.e., one neighbor in each axis. The FFAST decoder stops decoding if there is no more single-ton check node in any stage, i.e., all the check nodes are either zero-ton or multi-ton. Next, we find an upper bound on the probability of this 'bad' event.

Consider a set S of variable nodes such that $|S| = s$, where s is a small constant. Let E_S be an event that all the neighboring check nodes of all the variable nodes in the set S are multi-tons, i.e., if all the variable nodes except the ones in the set S are decoded, then, the FFAST decoder would fail to decode the set S. Also, let E be an event that there exists such a set. We first compute an upper bound on the probability of the event E_S, and then apply a union bound over all $\binom{k}{s}$ sets to get an upper bound on the probability of the event E.

Each variable node in the set S chooses an integer triplet (r_0, r_1, r_2) uniformly at random in a cube of size $\mathcal{P}_0 \times \mathcal{P}_1 \times \mathcal{P}_2$. Let $p_{\max}$ denote the maximum number of distinct values taken by these s variable nodes on any of the R_i axis. The FFAST decoder would stop decoding if and only if all the variable nodes have at least one neighbor along each of the 3 axes R_0, R_1, R_2 (see Fig. 3.8). This implies that $s \geq 4p_{\max}$. Also, $p_{\max} > 1$, hence $s \geq 8$, as by the CRT all the variable nodes s (with distinct associated integers) cannot have an identical triplet (r_0, r_1, r_2). An upper bound on the probability of the event E_S is then obtained as follows:

$$
\begin{aligned}
Pr(E_S) &< \prod_{i=0}^{2} \left(\frac{s}{4\mathcal{P}_i}\right)^s \binom{\mathcal{P}_i}{s/4} && (3.12)\\
&\approx \left(\frac{s}{4\mathbf{F}}\right)^{3s} \binom{\mathbf{F}}{s/4}^3 \\
&\overset{(a)}{<} \left(\frac{s}{4\mathbf{F}}\right)^{3s} \left(\frac{4\mathbf{F}e}{s}\right)^{3s/4} \\
&= \left(\frac{se^{1/3}}{4\mathbf{F}}\right)^{9s/4}, && (3.13)
\end{aligned}
$$

where in (a) we used $\binom{p}{q} \leq (pe/q)^q$. Then, using a union bound over all possible $\binom{k}{s}$ sets,

we get:

$$\begin{aligned} Pr(E) &< Pr(E_S)\binom{k}{s} \\ &< \left(\frac{se^{1/3}}{4\mathbf{F}}\right)^{9s/4}\left(\frac{ke}{s}\right)^{s} \\ &= O(1/n_b), \end{aligned} \tag{3.14}$$

where in the last inequality, we used $s \geq 8$, $k \propto \mathbf{F}^2$ and $n_b = O(\mathbf{F}^2)$.

Thus, the FFAST decoder decodes all the variable nodes with probability at least $1 - O(1/n_b)$.

3.6.2 Sketch of proof for $\delta = 1 - 1/d$, for integer $d \geq 3$

Consider a d-stage FFAST architecture with the following parameters. Let $\{\mathcal{P}_i\}_{i=0}^{d-1}$, be a set of pairwise co-prime integers such that $\mathcal{P}_i = \mathbf{F} + O(1),\ i = 0, 1, \ldots, d-1$, for some large integer $\mathbf{F}$. Set $k \propto \mathbf{F}^{d-1}$ and $n = \prod_{i=0}^{d-1} \mathcal{P}_i$, i.e., $k \propto n^{(d-1)/d}$. Also, let $f_i = \prod_{j=i}^{i+(d-2)} \mathcal{P}_{(j)_d}$, for $i = 0, \ldots, d-1$. The "expander-graph" property, of this construction for any value of d can be shown as follows:

- Decoding all the variable nodes: For $d = 3$, the worst case event is, the FFAST decoder fails to decode a set of size $|S| = 2^3 = 8$. For a general d, using an induction, one can show that the worst case failure event is when the FFAST decoder fails to decode a set of size $|S| = 2^d$. The probability of this event is upper bounded by $1/\mathbf{F}^{2^d - 2d}$.

3.6.3 Achieving the intermediate values of δ

In this section we show how to extend the scheme in Section 3.5, that was designed for $k \propto n^{1/3}$, to achieve a sparsity regime of $k \propto n^{(1+a)/(3+a)}$ for $a > 0$. This extension technique can be essentially used in conjunction with any of the operating points described earlier. Thus covering the full range of sparsity index $0 < \delta < 1$.

Choose a set of integer factors $\{\mathcal{P}_i\}_{i=0}^{3}$ that are pairwise co-prime and $\mathcal{P}_i = \mathbf{F} + O(1),\ i = 0, 1, 2$, while $\mathcal{P}_3 = \mathbf{F}^a + O(1)$ for some $a > 0$. Let $k \propto \mathbf{F}^{1+a}$ and $n = \prod_{i=0}^{3} \mathcal{P}_i$, i.e., $k \propto n^{(1+a)/(3+a)}$, and $f_0 = \mathcal{P}_0\mathcal{P}_3$, $f_1 = \mathcal{P}_1\mathcal{P}_3$ and $f_2 = \mathcal{P}_2\mathcal{P}_3$.

Union of Disjoint problems

The check nodes in each of the 3 sets are arranged so that a j^{th} check node in the set i, belongs to the row $(j)_{\mathcal{P}_i}$ and the column $(j)_{\mathcal{P}_3}$ (see Fig. 3.9). This is always possible using the CRT since $\{\mathcal{P}_i\}_{i=0}^{3}$ are pairwise co-prime and $f_i = \mathcal{P}_i\mathcal{P}_3,\ i = 0, 1, 2$.

A variable node with an associated integer v is uniquely represented by a quadruplet (r_0, r_1, r_2, r_3) where $r_i = (v)_{\mathcal{P}_i},\ i = 0, 1, 2, 3$ and is connected to the check node (r_i, r_3) in set

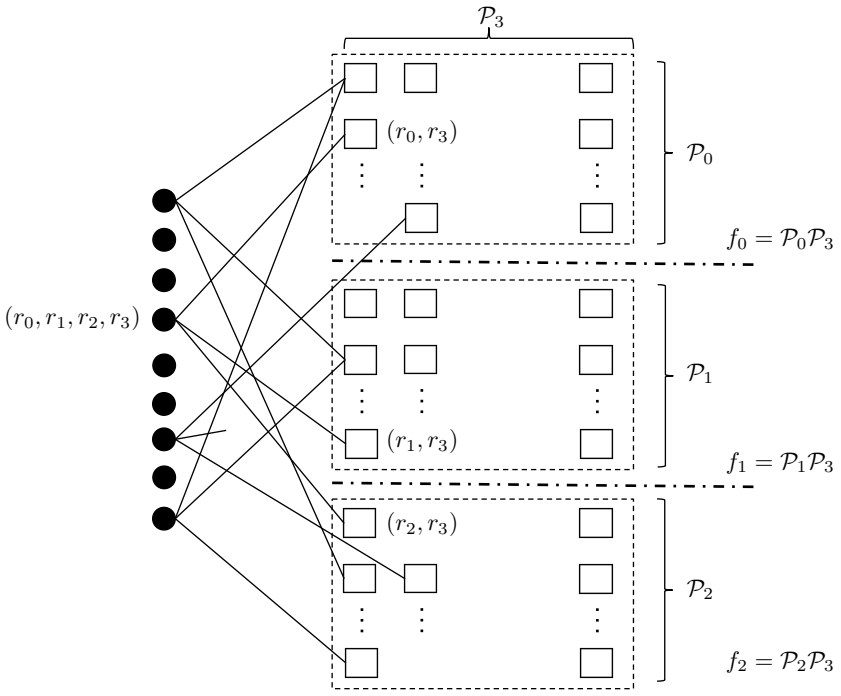


Figure 3.9: Let $\{\mathcal{P}_i\}_{i=0}^{3}$ be a set of pairwise co-prime integers, such that $\mathcal{P}_i \approx \mathbf{F}$, $i = 0, 1, 2$, and $\mathcal{P}_3 \approx \mathbf{F}^a$ for some $a > 0$. Also, let $k \propto \mathbf{F}^{1+a}$, $n = \prod_{i=0}^{3} \mathcal{P}_i$ and $f_0 = \mathcal{P}_0\mathcal{P}_3, f_1 = \mathcal{P}_1\mathcal{P}_3$ and $f_2 = \mathcal{P}_2\mathcal{P}_3$. The check nodes in each of the 3 sets are arranged so that a j^{th} check node in the set i belongs to the row $(j)_{\mathcal{P}_i}$ and the column $(j)_{\mathcal{P}_3}$. A variable node with an associated integer v is uniquely represented by a quadruplet (r_0, r_1, r_2, r_3) where $r_i = (v)_{\mathcal{P}_i}$, $i = 0, 1, 2, 3$, and is connected to the check node (r_i, r_3) in set i. Thus, the resulting bipartite graph is a union of $\mathcal{P}_3$ disjoint bi-partite graphs as shown in the figure.

i. Thus, the resulting bipartite graph is a union of $\mathcal{P}_3$ disjoint bi-partite graphs, where each bi-partite subgraph behaves as an instance of the 3-stage perfect co-prime case discussed in Section 3.5. Then, using a union bound over these disjoint graphs one can compute the probability of the FFAST decoder successfully decoding all but $o(k)$ and all the variable nodes for asymptotic values of k, n.

3.7 Sample and computational complexity of the FFAST algorithm

The FFAST algorithm performs the following operations in order to compute the n point DFT of an n-point discrete-time signal $\vec{x}$ (see Algorithm 3 in Section 3.4)

1. **Sub-sampling:** A FFAST architecture (see Fig. 3.1 in Section 3.1) with d stages, has d distinct subsampling patterns chosen as per the discussions in Sections 3.5 and 3.6. These patterns are deterministic and are pre-computed. We assume the presence of random-access-memory, with unit cost per I/O operation, for reading the subsamples. For the very-sparse regime ($k \propto n^{\delta},\ 0 < \delta \leq 1/3$) as shown in Section 3.5, the FFAST

architecture has $d = 3$ stages and 2 sub-streams per stage. Each sub-stream has approximately $\mathbf{F} = \eta k$ number of input samples. Hence, the total number of samples used for the very-sparse regime is $m = 6\eta k = 2.44k$ (see Section 3.5 Table 3.4). For the less-sparse regime, choice of the FFAST architecture parameters is bit more involved and depends on the sparsity index δ. A $d \leq 8$ stage FFAST architecture, in conjunction with discussion of Section 3.6.3, is sufficient to achieve the sparsity index of $1/3 < \delta < 0.99$. Thus, again using the values from Table 3.4 of Section 3.5 for $d = 8$, for the less-sparse regime of $1/3 < \delta < 0.99$ the number of samples $m \leq 3.74k$. In general for any fixed $0 < \delta < 1$, the sample complexity m can be as small as rk, where r is a constant that depends on the sparsity index δ.

2. **DFT:** The FFAST algorithm computes $2d$ number of DFT's, each of length approximately equal to $\mathbf{F} = \eta k$, of the subsampled input data corresponding to 2 sub-streams in each of the d stages. Using a standard FFT algorithm, e.g., prime-factor FFT or Winograd FFT algorithm [4], one can compute each of these DFT's in $O(k \log k)$ computations. Thus, the total computational cost of this step is $O(k \log k)$.

3. **Peeling-decoder over sparse graph codes:** It is well known [53], that the computational complexity of the peeling-decoder over sparse graph codes is linear in the dimension of the graph, i.e., $O(k)$.

Thus, the FFAST algorithm computes a k-sparse n-point DFT with $O(k)$ samples using no more than $O(k \log k)$ arithmetic operations for all $0 < \delta < 1$.

Note, that the FFAST architecture and the associated peeling-decoder operations are highly parallelizable, which can provide significant speed improvement in terms of real time performance.

3.8 Simulation Results

In this section we validate the empirical performance of our FFAST algorithm for a wide variety of signals having an exactly sparse Fourier spectrum. We show the performance of the FFAST algorithm for two sparsity regimes, what we call the *very-sparse regime* and the *less-sparse regime*. We contrast the observed empirical performance, in terms of various metrics like threshold behavior of *density evolution*, iterations, sample complexity and computational complexity, with the theoretical claims of Theorem 3.3.1.

3.8.1 The CRT based graph ensemble behaves like the balls-and-bins based graph ensemble

In this section we empirically show that the CRT based constructions show a very sharp threshold behavior which is in close agreement with the theoretical claims in this chapter.

Regimes	Signal dimension n	Sparsity k	$k = n^{\delta}$, δ	stages d	$m \approx 2d\eta k$		ℓ	failures
					m	η		
Very-sparse	511*512*513 $\approx 134 \times 10^6$	900	0.363	3	3072	0.569	6	1
		1000	0.369		3072	0.512	9	0
		1100	0.374		3072	0.465	13	1
		1200	0.378		3072	0.427	18	99
Less-sparse	16*17*19*21 $\approx 0.1 \times 10^6$	13000	0.81	4	48094	0.462	6	0
		15000	0.83		48094	0.401	8	0
		17000	0.84		48094	0.354	13	2
		19000	0.85		48094	0.316	29	10000

Table 3.5: Shows the performance of the FFAST algorithm for two different sparsity regimes: 1) Very-sparse regime: $k \propto n^{1/3}$. For this regime, a $d = 3$ stage FFAST architecture is chosen. The number of samples per sub-stream in each of the 3 stages are perfectly co-prime: $f_0 = 511, f_1 = 512$ and $f_2 = 513$ respectively, and 2) Less-sparse regime: $n^{0.73} < k < n^{0.85}$. For this regime, a $d = 4$ stage FFAST architecture is chosen. The number of samples per sub-stream in each of the 4 stages are *not* co-prime but have "cyclically-shifted" overlapping co-prime factors: $f_0 = 16 \times 17 \times 19 = 5168$, $f_1 = 17 \times 19 \times 21 = 6783$, $f_2 = 19 \times 21 \times 16 = 6384$ and $f_3 = 21 \times 16 \times 17 = 5712$ respectively. We note that the FFAST algorithm exhibits a threshold behavior in terms of sample complexity m. For example, when $\eta_1 \geq 0.427$ and $\eta_2 \geq 0.354$, for the very-sparse and the less-sparse regimes respectively, the FFAST algorithm successfully computes all the non-zero DFT coefficients for almost all the runs. Further, in one or two instances when it failed to recover all the non-zero DFT coefficients, it has recovered all but 8 (for $d = 3$) or 16 (for $d = 4$) non-zero DFT coefficients. This validates our theoretical findings of the bottleneck failure event being that the FFAST decoder decodes all but a handful (2^d) of the DFT coefficients. From Table 3.4 in Section 3.5.3, we see that the optimal threshold values for the very-sparse and less-sparse regimes are $\eta_1^* = 0.4073$ and $\eta_2^* = 0.3237$ resp., which are in close agreement with the simulation results. The table also shows that the average number of iterations ℓ, required for the FFAST algorithm to successfully compute the DFT for both the regimes, are quite modest.

Thus, confirming that the CRT ensemble is equivalent to the balls-and-bins ensemble, for both the 'very-sparse' and the 'less-sparse' regimes, as was used in the theoretical analysis.

Simulation Setup

- **Very sparse regime:** The desired signal $\vec{x}$ is of ambient dimension $n = 511 \times 512 \times 513 \approx 134 \times 10^6$. The sparsity parameter k is varied from 400 to 1300 which corresponds to the very-sparse regime of $k \propto n^{1/3}$. We consider a FFAST architecture with $d = 3$ stages and 2 sub-streams per stage, i.e., two delay chains per stage (see Fig. 3.1). The uniform sampling periods for the 3 stages are 512×513, 511×513 and 511×512 respectively. This results in the number of samples per sub-stream, for the three stages to be $f_0 = 511$, $f_1 = 512$ and $f_2 = 513$ respectively. Note that the number of samples per sub-stream in all the three different stages, i.e., f_i's, are pairwise co-prime. The total number of samples used by the FFAST algorithm for this simulation is[6] $m < 2(f_0 + f_1 + f_2) = 3072$.

[6]The samples used by the different sub-streams in the three different stages overlap, e.g., $x[0]$ is common to all the zero delay sub-streams in each stage. Hence, $m < 2(f_0 + f_1 + f_2)$ and not equal.

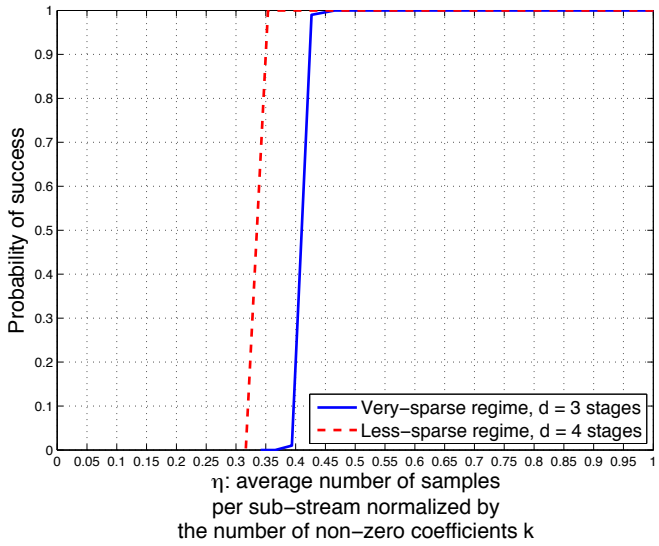


Figure 3.10: The probability of success of the FFAST algorithm as a function of η, the average number of samples per sub-stream normalized by the number of non-zero coefficients k. The plot is obtained for the two different sparsity regimes: 1) Very-sparse regime, i.e., $k \propto n^{1/3}$. For this regime, a $d = 3$ stage FFAST architecture is chosen. The number of samples per sub-stream in the three stages are perfectly co-prime, ($f_0 = 511, f_1 = 512$ and $f_2 = 513$), and 2) Less-sparse regime, specifically ($n^{0.73} < k < n^{0.85}$). For this regime, a $d = 4$ stage FFAST architecture is chosen. The number of samples per sub-stream in the 4 stages are not co-prime, ($f_0 = 5168, f_1 = 6783, f_2 = 6384$ and $f_3 = 5712$). Each point on the plot is obtained by averaging over 10000 runs. The ambient signal dimension n and the number of samples m are fixed, while the number of non-zero coefficients k is varied to get different values of η. We note that the FFAST algorithm exhibits a threshold behavior with $\eta_1 = 0.427$ being the threshold for the very-sparse regime, and $\eta_2 = 0.354$ for the less-sparse regime respectively. From Table 3.4 in Section 3.5.3, we see that the optimal threshold values are $\eta_1^* = 0.4073$ and $\eta_2^* = 0.3237$, which are in close agreement with our simulation results.

- **Less sparse regime:** The desired signal $\vec{x}$ is of ambient dimension $n = 16 \times 17 \times 19 \times 21 \approx 0.1 \times 10^6$. The sparsity parameter k is varied from 5000 to 19000 which corresponds to the less-sparse regime of $n^{0.73} < k < n^{0.85}$. We consider a FFAST architecture with $d = 4$ stages and 2 sub-streams per stage. The uniform sampling periods for the 4 stages are 21, 16, 17 and 19 respectively. This results in the number of samples per sub-stream, for the four stages to be $f_0 = 16 \times 17 \times 19 = 5168$, $f_1 = 17 \times 19 \times 21 = 6783$, $f_2 = 19 \times 21 \times 16 = 6384$ and $f_3 = 21 \times 16 \times 17 = 5712$ respectively. Note that the number of samples per sub-stream in the four stages are composed of "cyclically-shifted" co-prime numbers and are not pairwise co-prime. The total number of samples used by the FFAST algorithm for this simulation is $m < 2(f_0 + f_1 + f_2 + f_3) = 48094$.

- For each run, an n-dimensional k-sparse signal $\vec{X}$ is generated with non-zero values $X_i \in \{\pm 10\}$ and the positions of the non-zero coefficients are chosen uniformly at random in $\{0, 1 \ldots, n-1\}$. The time-domain signal $\vec{x}$ is then generated from $\vec{X}$ using an IDFT operation. This discrete-time signal $\vec{x}$ is then given as an input to our FFAST architecture.

- Each sample point in the Fig. 3.10, is generated by averaging over 10000 runs.
- Decoding is successful if all the DFT coefficients are recovered perfectly.

We observe the following aspects of the simulations in detail and contrast it with the claims of Theorem 3.3.1

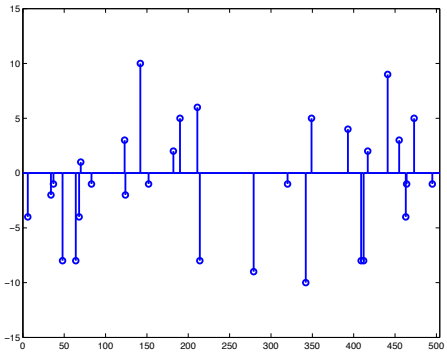
Density Evolution threshold η The density evolution recursion (3.8) in Section 3.5.3, implies a threshold behavior: if the average number of samples per sub-stream normalized by k, i.e., η, is above a certain threshold (as specified in Section 3.5.3 Table 3.4), then the recursion (3.8) converges to 0 as $\ell \to \infty$ otherwise p_ℓ is strictly bounded away from 0. In Fig. 3.10 we plot the probability of success, averaged over 10000 runs, of the FFAST algorithm as a function of η, i.e., varying k for a fixed number of samples m. We note that the FFAST algorithm exhibits a threshold behavior with $\eta_1 = 0.427$ being the threshold for the very-sparse regime with $d = 3$, and $\eta_2 = 0.354$ for the less-sparse regime with $d = 4$ respectively. From Table 3.4 in Section 3.5.3, we see that the optimal threshold values are $\eta_1^* = 0.4073$ and $\eta_2^* = 0.3237$, which are in close agreement with our simulation results of Fig. 3.10.

The FFAST algorithm is verified to compute the DFT with high probability using as few as $m = rk$ samples, where the oversampling ratio $r \approx 2d\eta < 4$ as given in Table 3.4.

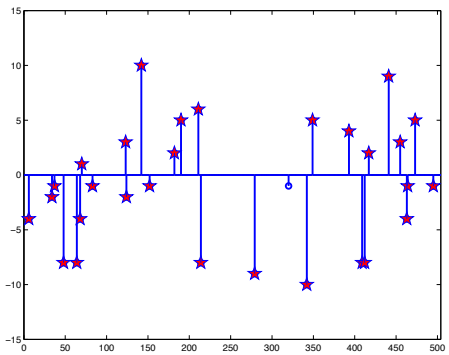
Iterations In the theoretical analysis of Section 3.5.3, we showed that the FFAST algorithm, if successful, decodes all the DFT coefficients in ℓ iterations, where, ℓ is a large but a fixed constant. In order to get an empirical sense of how large ℓ has to be, we consider an example with $n = 504$, $k = 30$, and a three-stage FFAST architecture. The number of samples per sub-stream in the three stages are $f_0 = 56, f_1 = 72$ and $f_2 = 63$ respectively. Further, each stage has 2 delay sub streams. An example of an n-length input signal $\vec{x}$ that has a 30-sparse DFT is shown in Fig. 3.11(a). In this simulation, we observe that the FFAST algorithm computes the sparse DFT $\vec{X}$ perfectly in $\ell = 2$ iterations, as shown in Fig 3.11(c) using $m = 2\sum_{i=0}^{2} f_i = 382$ samples of $\vec{x}$. Note that the number of samples $m > 4k$ for this example. This is done intentionally for the illustrative purpose and to avoid the clutter in Fig. 3.11. Table 3.5 shows more empirical values of ℓ for different settings of the problem.

3.8.2 Sample and Computational Complexity

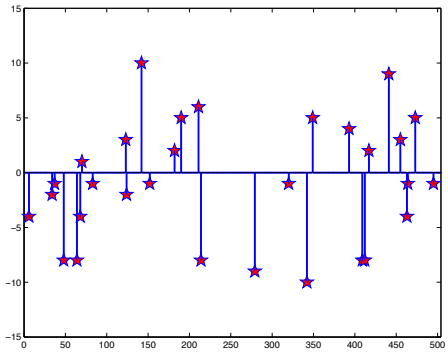
In this section, we empirically validate the sample and the computational complexity of the FFAST algorithm. In particular, we show that both the sample and the computational complexity are independent of the signal dimension n and are given by $m = rk$, where oversampling ratio $r < 4$, and $O(k \log k)$ respectively.



(a) Original k-sparse n-length DFT $\vec{X}$, where $k = 30$ and $n = 504$.



(b) Estimated signal $\hat{\vec{X}}$ after 1 iteration.

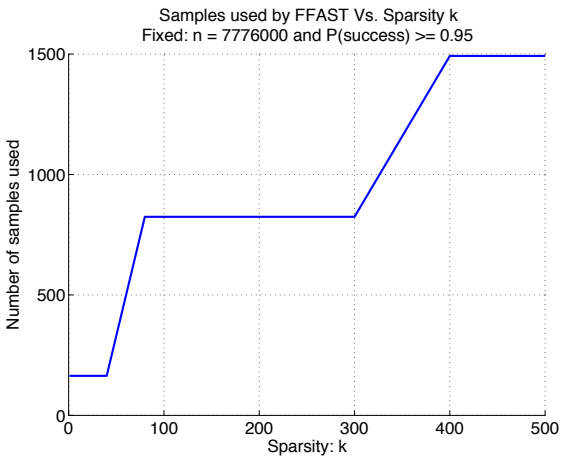


(c) Estimated signal $\hat{\vec{X}} = \vec{X}$ after 2 iterations.

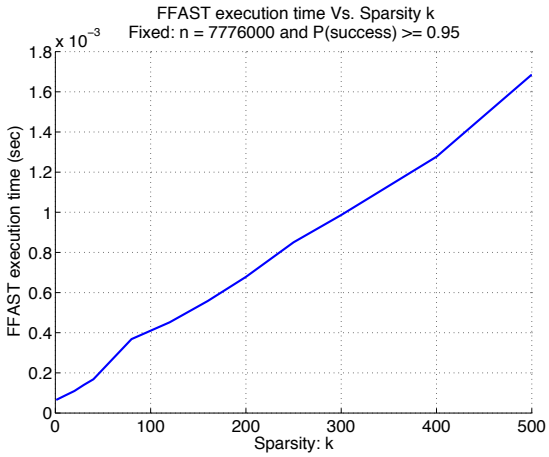
Figure 3.11: Consider a FFAST architecture with $d = 3$ stages, ambient signal dimension $n = 504$ and sparsity $k = 30$. The number of samples per sub-stream in the three stages are $f_0 = 56, f_1 = 72$ and $f_2 = 63$ respectively. Further, each stage has 2, delay sub-streams. Thus, the total number of samples $m = 2\sum_{i=0}^{2} f_i = 382$. Note that the number of samples $m > 4k$ for this example. This is done intentionally for the illustrative purpose and to avoid the clutter in the figure. The figure shows a) the original 30-sparse spectrum $\vec{X}$. b) the spectrum estimated by the FFAST algorithm after 1 iteration. Note that it has correctly recovered 29 out of the 30 non-zero DFT coefficients. c) The FFAST algorithm recovers all the 30 non-zero DFT coefficients perfectly after 2 iterations.

Dependency on k We consider a signal $\vec{x}$ of length $n \approx 7.7 \times 10^6$ that has a k-sparse DFT $\vec{X}$. We vary the sparsity parameter k from 10 to 500. For each sparsity, we design a $d = 3$ stage FFAST architecture.

In Fig. 3.12(a) we plot the number of samples m required for the FFAST algorithm to successfully reconstruct all the non-zero DFT coefficients for different values of k. We note that the plot is piece-wise flat, since the design space of the sub-sampling patterns is not continuous. The oversampling ratio $r \approx 3$ at the highest sparsity level for every flat segment of the plot. Thus, justifying the theoretical claim of $r < 4$ for asymptotically large values of k, n. The plot in Fig. 3.12(b) shows the average time in sec required by the FFAST



(a) Number of samples used by the FFAST algorithm for varying values of sparsity.



(b) Average execution time, i.e., computational complexity, of the FFAST algorithm to successfully reconstruct all the non-zero DFT coefficients.

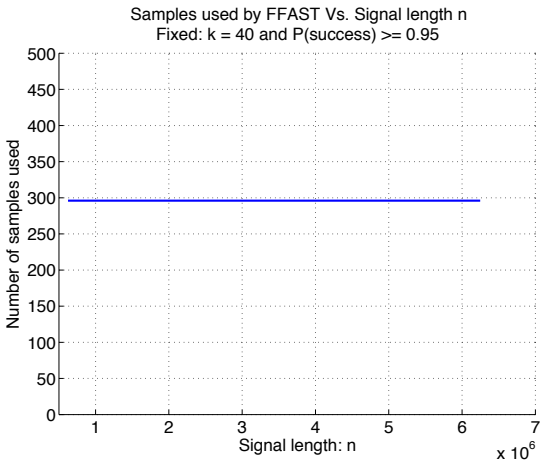
Figure 3.12: Consider a signal $\vec{x}$ of length $n \approx 7.7 \times 10^6$ that has a k-sparse DFT $\vec{X}$. We vary the sparsity parameter k from 10 to 500. For each sparsity we design a $d = 3$ stage FFAST architecture. Each point in both the plots is generated by averaging over 200 runs. A successful run is when the FFAST algorithm recovers all the non-zero DFT coefficients. Probability of success > 0.95 means that parameters are chosen so that the FFAST algorithm succeeds in 95% or more runs the simulations.

algorithm, for a successful reconstruction of the DFT $\vec{X}$. We note that the increase in time is almost super-linear, i.e., $O(k \log k)$, in the sparsity k of the signal.

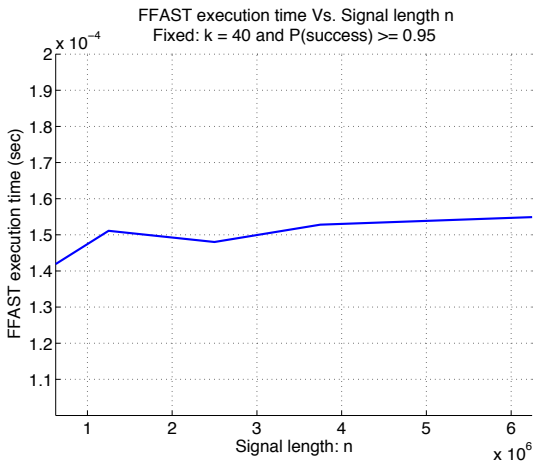
Dependency on n Consider a signal $\vec{x}$ of length n that has a $k = 40$ sparse DFT $\vec{X}$. We vary the ambient dimension n of the signal from 0.6 million to 6 million keeping the sparsity fixed at 40. In Fig. 3.13(a) and Fig. 3.13(b) respectively, we plot the number of samples used and the average execution time required for the FFAST algorithm to successfully reconstruct all the $k = 40$ non-zero DFT coefficients. Note that both the sample complexity and the computational complexity are almost constant even when the signal length n changes by a ten fold factor.

3.A Edge degree-distribution polynomial for balls-and-bins model

The edge-degree distribution polynomial of a bi-partite graph is defined as $\rho(\alpha) = \sum_{i=1}^{\infty} \rho_i \alpha^{i-1}$, where ρ_i denotes the probability of an edge (or a fraction of edges) of the graph is connected to a check node of degree i. Recall that in the randomized construction based on a balls-and-bins model described in Section 3.5.1, every variable node chooses one neighboring node



(a) Number of samples used by the FFAST algorithm for varying values of signal length n.



(b) Average execution time, i.e., computational complexity, of the FFAST algorithm to successfully reconstruct all the non-zero DFT coefficients.

Figure 3.13: Consider a signal $\vec{x}$ of length n that has a $k = 40$ sparse DFT $\vec{X}$. We vary the ambient dimension n of the signal from 0.6 million to 6 million (by factor of 10) keeping the sparsity fixed at 40. Note that both the sample complexity and the computational complexity are almost constant even when the signal length n changes by a ten fold factor.

in each of the d subsets of check nodes uniformly at random. Thus, the number of edges connected to a check node, in the subset with f_0 check nodes, is a binomial $B(1/f_0, k)$ random variable. So when k is large and $f_0 = \eta k + O(1)$, the binomial distribution $B(1/f_0, k)$ is well approximated by a Poisson random variable with mean $1/\eta$. Thus,

$$Pr(\text{check node has edge degree} = i) \approx \frac{(1/\eta)^i e^{-1/\eta}}{i!}. \tag{3.15}$$

Let $\rho_{0,i}$ be the fraction of the edges, that are connected to a check node of degree i in set 0. Then, we have,

$$\begin{aligned} \rho_{0,i} &= \frac{if_0}{k} Pr(\text{check node has edge degree} = i) \\ &\overset{(a)}{\approx} \frac{if_0}{k} \frac{(1/\eta)^i e^{-1/\eta}}{i!} \\ &\overset{(b)}{\approx} \frac{(1/\eta)^{i-1} e^{-1/\eta}}{(i-1)!}, \end{aligned} \tag{3.16}$$

where (a) follows from Poisson approximation of the Binomial random variable and (b) from $f_0 = \eta k + O(1)$. Since, $f_i = \eta k + O(1)$ for all $i = 0, 1 \ldots, d-1$. We have,

$$\rho_i \approx \frac{(1/\eta)^{i-1} e^{-1/\eta}}{(i-1)!} \tag{3.17}$$

and

$$\rho(\alpha) = e^{-(1-\alpha)/\eta} \tag{3.18}$$

3.B Proof of Lemma 3.5.4

Proof. We provide a proof using a contradiction argument. If possible let S be the set of the variable nodes that the peeling-decoder fails to decode. We have $|S| \leq \alpha k$. Without loss of generality let $|N_1(S)| \geq |N_i(S)|,\ \forall i \in \{0, \ldots, d-1\}$. Then, by the hypothesis of the Theorem $|N_1(S)| > |S|/2$.

Note that the peeling-decoder fails to decode the set S if and only if there are no more singleton check nodes in the neighborhood of S and in particular in $N_1(S)$. For all the check nodes in $N_1(S)$ to be multi-ton, the total number of edges connecting to the check nodes in set $N_1(S)$ have to be at least $2|N_1(S)| > |S|$. This is a contradiction since there are only $|S|$ edges going from set S to $N_1(S)$ by construction. □

3.C Proof of Lemma 3.5.5

Proof. Consider a set S of variable (left) nodes in a random graph from the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$, where $|\mathbf{F}| = d \geq 3$. Let $N_i(S)$ be the right neighborhood of the set S in the i^{th} subset of check nodes, for $i = 0, 1, \ldots, d-1$. Also, let E_S denote the event that the all the d right neighborhoods of S are of size $|S|/2$ or less, i.e., $\max\{|N_i(S)|\}_{i=0}^{d-1} \leq |S|/2$. First, we compute an upper bound on the probability of the event E_S as follows:

$$\begin{aligned} Pr(E_S) &< \prod_{i=0}^{d-1} \left(\frac{|S|}{2f_i}\right)^{|S|} \binom{f_i}{|S|/2} \\ &\overset{(a)}{\approx} \left(\frac{|S|}{2\mathbf{F}}\right)^{d|S|} \binom{\mathbf{F}}{|S|/2}^{d} \\ &< \left(\frac{|S|}{2\mathbf{F}}\right)^{d|S|} \left(\frac{2\mathbf{F}e}{|S|}\right)^{d|S|/2} \\ &< \left(\frac{|S|e}{2\mathbf{F}}\right)^{d|S|/2} \end{aligned} \tag{3.19}$$

where the approximation (a) uses $f_i = \mathbf{F} + O(1)$ for all $i = 0, \ldots, d-1$. Next, using a union bound, over all possible sets of size $|S|$, we get an upper bound on the probability of an event E, that there exists some set of variable nodes of size $|S|$, whose all the d right

neighborhoods are of size $|S|/2$ or less.

$$\begin{aligned} Pr(E) &< Pr(E_S)\binom{k}{|S|} \\ &< \left(\frac{|S|e}{2\mathbf{F}}\right)^{d|S|/2}\left(\frac{ke}{|S|}\right)^{|S|} \\ &\overset{(b)}{<} \left[\left(\frac{|S|}{\mathbf{F}}\right)^{d-2}\left(\frac{e}{2}\right)^{d}\left(\frac{e}{\eta}\right)^{2}\right]^{|S|/2} \\ &< O\left((|S|/n_b)^{|S|/2}\right) \end{aligned} \tag{3.20}$$

where in (b) we used $\mathbf{F} = \eta k$ and in the last inequality we have used $d \geq 3$ and $n_b = O(\mathbf{F})$. Then, specializing the bound in (3.20) for $|S| = \alpha k$, for small enough $\alpha > 0$, and $|S| = o(k)$ we get,

- For $|S| = \alpha k$, for small enough $\alpha > 0$:

$$Pr(E) < e^{-\epsilon k \log(n_b/k)}, \quad \text{for some} \quad \epsilon > 0 \tag{3.21}$$

- For $|S| = o(k)$:

$$Pr(E) < O\left(1/n_b\right) \tag{3.22}$$

□

3.D Proof of Lemma 3.5.2

Proof. a) [Expected behavior] Consider decoding on a random graph in the ensemble $\mathcal{C}_1^k(\mathcal{F}, n_b)$. Let Z_i, $i = 0, \ldots, kd-1$, be an indicator random variable that takes value 1 if the edge $\vec{e}_i$ is not decoded after ℓ iterations of the peeling-decoder and 0 otherwise. Then by symmetry $\mathbb{E}[Z] = kd\mathbb{E}[Z_1]$. Next, we compute $\mathbb{E}[Z_1]$ as,

$$\begin{aligned} \mathbb{E}[Z_1] &= \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}]Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}) \\ &\quad + \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}]Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}) \\ &\leq \mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is tree-like}] + Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}). \end{aligned}$$

In Appendix 3.E we show that $Pr(\mathcal{N}_{\vec{e}_1}^{2\ell} \text{ is not tree-like}) \leq \gamma/\mathbf{F}$ for some positive constant γ. Also we have $\mathbb{E}[Z_1 \mid \mathcal{N}_{\vec{e}_1}^{2\ell} \text{is tree-like}] = p_\ell$ by definition. Thus,

$$\mathbb{E}[Z] < kdp_\ell + \frac{d\gamma}{\eta}, \tag{3.23}$$

where we have used $\mathbf{F} = \eta k$.

b) [Concentration] In this part, we want to show that the number of edges Z, that are not decoded at the end of ℓ iterations of the peeling-decoder, is highly concentrated around kdp_ℓ. There are two potential ways of an edge not being decoded after ℓ iterations: 1) the 2ℓ depth neighborhood of the edge is not tree-like or 2) the neighborhood is tree-like but the edge belongs to the fraction of edges that are not decoded after ℓ rounds (as per density evolution equation (3.8)). Let T be the number of edges, out of the total of kd, that have tree-like neighborhood of depth 2ℓ. Let $T' \subseteq T$, be the number of edges not decoded despite having tree-like neighborhood. Then from Appendix 3.E we know,

$$\mathbb{E}[T] \;>\; kd(1-\epsilon/4) \tag{3.24}$$

for any $\epsilon > 0$. Moreover, from equation (3.8)

$$\mathbb{E}[T'] \;=\; Tp_\ell. \tag{3.25}$$

Next, we obtain a concentration result for T, by using the (now) standard argument of exposing the k variable nodes v_j, $j = 1, \ldots k$, (and hence the edges in the graph) one by one to set up a Doob's martingale and applying Azuma's inequality. In particular, let $Y_i = \mathbb{E}[T|v_1^i]$ be the expected value of T after exposing i variable nodes. Then, $Y_0 = \mathbb{E}[T]$ and $Y_k = T$ and the sequence Y_i forms a Doob martingale.

Now, consider any pair of graphs (determined by the choice of variable nodes) that differ only on $(i+1)^{th}$ variable node. The number of edges with tree-like neighborhood of depth 2ℓ in these graphs differ at most by a constant number, since the differing variable node can influence the 2ℓ depth neighborhood of only constant number of edges (since the left edge degree d is a constant). Hence, $|Y_{i+1} - Y_i| < c_i$, for some constant $c_i \;\forall\; i = 0, \ldots, k-1$. Hence, using Azuma's inequality along with (3.24), we get,

$$Pr\left(kd - T > \epsilon kd/2\right) < \exp(-\beta_1 \epsilon^2 k), \tag{3.26}$$

for some constant β_1 that depends on the left degree d and constant η. Using Azuma's inequality the following concentration result for T' holds,

$$Pr\left(|T' - Tp_\ell| > \epsilon kd/2\right) < 2\exp(-\beta_2 \epsilon^2 k). \tag{3.27}$$

The assertion then follows from (3.26), (3.27) and $T' \leq Z \leq T' + |kd - T|$. □

3.E Probability of Tree-like Neighborhood

Consider an edge $\vec{e}$ in a randomly chosen graph $\mathcal{G} \in \mathcal{C}_1^k(\mathcal{F}, n_b)$. Next, we show that the neighborhood $\mathcal{N}_{\vec{e}}^{2\ell^*}$ of $\vec{e}$ is tree-like with high probability, for any fixed ℓ^*. Towards that end, we first assume that the neighborhood $\mathcal{N}_{\vec{e}}^{2\ell}$ of depth 2ℓ, where $\ell < \ell^*$, is tree-like and

show that it remains to be tree-like when extended to depth $2\ell+1$ w.h.p. Let $C_{\ell,i}$ be the number of check nodes, from set i, and M_ℓ be the number of variable nodes, present in $\mathcal{N}_{\vec{e}}^{2\ell}$. Also assume that t more edges from the leaf variable nodes in $\mathcal{N}_{\vec{e}}^{2\ell}$ to the check nodes at depth $2\ell+1$ are revealed without creating a loop. Then, the probability that the next revealed edge from a leaf variable node to a check node (say in set i) does not create a loop is $\frac{f_i - C_{\ell,i} - t}{f_i - C_{\ell,i}} \geq 1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like, given $\mathcal{N}_{\vec{e}}^{2\ell}$ is tree-like, is lower bounded by $\min_i (1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}})^{C_{\ell+1,i} - C_{\ell,i}}$. Similarly assume that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like and s more edges from check nodes to the variable nodes at depth $2\ell+2$ are revealed without creating a loop. Note the probability that a check node has degree ≥ 2 is upper bounded by $k/\mathbf{F}$ and conditioned on the event that a check node has an outgoing edge it has equal chance of connecting to any of the edges of the variable nodes that are not yet connected to any check node. Thus, the probability of revealing a loop creating edge from a check node to a variable node at depth $2\ell+2$ is upper bounded by, $(k/\mathbf{F})(1 - \frac{(k - M_\ell - s)d}{kd - M_\ell d - s}) \leq \frac{kM_{\ell^*}}{\mathbf{F}(k - M_{\ell^*})}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+2}$ is tree-like given $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is tree-like is lower bounded by $(1 - \frac{kM_{\ell^*}}{\mathbf{F}(k - M_{\ell^*})})^{M_{\ell+1} - M_\ell}$.

It now follows that the probability that $\mathcal{N}_{\vec{e}}^{2\ell^*}$ is tree-like is lower bounded by

$$\min_i \left(1 - \frac{kM_{\ell^*}}{\mathbf{F}(k - M_{\ell^*})}\right)^{M_{\ell^*}} \left(1 - \frac{C_{\ell^*,i}}{f_i - C_{\ell^*,i}}\right)^{C_{\ell^*,i}}$$

Hence, for k sufficiently large and fixed ℓ^*,

$$Pr\left(\mathcal{N}_{\vec{e}}^{2\ell^*} \text{not tree-like}\right) \leq \max_i \frac{M_{\ell^*}^2}{\mathbf{F}} + \frac{C_{\ell^*,i}^2}{f_i} \leq \frac{\gamma}{\mathbf{F}}$$

for some constant $\gamma > 0$, since for any fixed value of ℓ^*, $C_{\ell^*,i}$ and M_{ℓ^*} are constant w.h.p and $f_i = \mathbf{F} + O(1)$ for all $i = 0, \ldots, d-1$.

Chapter 4

Stable recovery of approximately sparse DFT

4.1 Introduction

The Fast Fourier Transform (FFT) is the fastest known way to compute the DFT of an arbitrary n-length signal, and has a computational complexity of $O(n \log n)$[1]. Many applications of interest involve signals, e.g. relating to audio, image, and video data, seismic signals, biomedical signals, financial data, social graph data, cognitive radio applications, surveillance data, satellite imagery, etc., which have a sparse Fourier spectrum. In such cases, a small subset of the spectral components typically contains most or all of the signal energy, with most spectral components being either zero or negligibly small. In Chapter 3, we have proposed a novel FFAST (Fast Fourier Aliasing-based Sparse Transform) framework, that exploits this additional sparse structure to compute the n-length DFT $\vec{X}$, using only $O(k)$ time-domain samples in $O(k \log k)$ arithmetic computations, for the case when the n-length DFT, $\vec{X}$, has exactly k non-zero coefficients, where $k << n$. For long signals, i.e., when n is of order of millions or tens of millions, as is becoming more relevant in the Big-data age, the gains over conventional FFT algorithms can be significant.

In contrast, although many real world signals have a compact representation in the spectral-domain, they are rarely *exactly sparse*, i.e., have precisely k non-zero DFT coefficients. Typically such signals can be modeled as approximately sparse signals. Moreover, the observations acquired through typical sensing mechanisms have imperfections, e.g., the acquired samples may have additional thermal noise at the acquisition sensors, or have imperfect sampling due to jitter, sampling pulse shape, quantization effects, etc. The idealized

[1]Recall that a single variable function $f(x)$ is said to be $O(g(x))$, if for a sufficiently large x the function $|f(x)|$ is bounded above by $|g(x)|$, i.e., $\lim_{x\to\infty} |f(x)| < c|g(x)|$ for some constant c. Similarly, $f(x) = \Omega(g(x))$ if $\lim_{x\to\infty} |f(x)| > c|g(x)|$ and $f(x) = o(g(x))$ if the growth rate of $|f(x)|$ as $x \to \infty$, is negligible as compared to that of $|g(x)|$, i.e. $\lim_{x\to\infty} |f(x)|/|g(x)| = 0$.

assumptions in Chapter 3, of the signal being exactly sparse and the observations being perfect, were used primarily to highlight the novel ideas underlying the FFAST architecture with conceptual clarity.

In this chapter, we extend the FFAST framework of Chapter 3 to address the effects of imperfect measurements due to additional thermal noise at the acquisition sensors, i.e., noise in the observations. In particular, we show that the FFAST framework acquires and reconstructs an n-length signal $\vec{x}$, whose DFT $\vec{X}$ has k non-zero coefficients, using $O(k \log^2 k \log n)$ noise-corrupted samples, in $O(n \log n \log^2 k)$ computations. This contrasts with the best known scaling of the partial Fourier measurements, i.e., $O(k \log^2 k \log n)$, in compressed-sensing literature [66]. Our algorithm succeeds with probability approaching 1 asymptotically in the number of measurements. We emphasize the following caveats. First, we assume that the non-zero DFT coefficients of the signal $\vec{x}$ have uniformly random support and take values from a finite constellation[2](as explained in Section 4.2). Secondly, our results are probabilistic and are applicable for asymptotic values of k, n, where k is sub-linear in n. Lastly, we assume i.i.d Gaussian noise model for observation noise.

The sparse signal acquisition framework and the associated peeling-based reconstruction algorithm SWIFT (Short-and-Wide-Iterative-Fast-Transform) discussed in Chapter 2 are very similar to the FFAST front-end architecture and the FFAST peeling-decoder. Hence, all the analysis techniques and the results developed in this chapter carry over in a straightforward way to the discrete compressed sensing problem of Chapter 2.

4.1.1 Main idea

We use a simple example to illustrate the key ideas. Consider an $n = 20$ length input signal $\vec{x}$, whose DFT $\vec{X}$, is $k = 5$ sparse. Further, let the 5 non-zero DFT coefficients of $\vec{x}$ be $X[1]$, $X[3]$, $X[5]$, $X[10]$ and $X[13]$. Let $\vec{y} = \vec{x} + \vec{w}$ be the noise-corrupted observation of the signal. In general, the FFAST sub-sampling 'front-end' consists of d stages, where each stage further has D subsampling paths. For illustrative purposes, in Fig. 4.1 we show the processing of $\vec{y}$ through stage 0 of a 2-stage FFAST architecture. The FFAST front-end subsamples 3 circularly shifted versions of the observation $\vec{y}$, by a sampling period of 5. The output of stage 0, of the FFAST front-end, is then obtained by computing the 4-point DFT of each of the sub-sampled stream and further grouping them into 'bins', as shown in Fig. 4.1. Let $\vec{y}_{b,i,j}$ denote the 3-dimensional observation vector of bin j of stage i. Using the basic signal processing identities of sampling-aliasing and circular shifts, the relation between the

[2]The framework and the reconstruction algorithm generalize to arbitrary complex-valued non-zero DFT coefficients as long as all the non-zero DFT coefficients are above a certain threshold. But the proof technique becomes much cumbersome and we do not address this issue in this chapter.

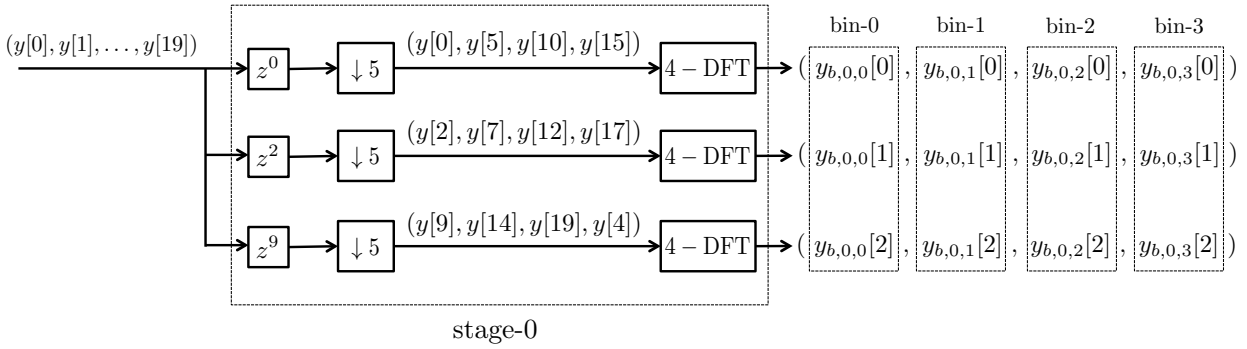


Figure 4.1: The noise-corrupted observation $\vec{y} = \vec{x} + \vec{w}$, is processed through a 2-stage FFAST architecture. In general, the sub-sampling front-end of the FFAST architecture consists of 3 or more stages depending on the sparsity index δ, where $k = O(n^{\delta})$. In this example, we consider the FFAST architecture with 2-stages only for the purpose of illustration. Further, here we show only stage 0 of a 2-stage FFAST architecture. The 20-point DFT of the signal $\vec{x}$ is 5 sparse, with the non-zero DFT coefficients being $X[1]$, $X[3]$, $X[5]$, $X[10]$ and $X[13]$. The FFAST front-end subsamples 3 circularly shifted versions of the observation $\vec{y}$, by a sampling period of 5. The number and the pattern of the circular shifts are chosen carefully as explained in Section 4.5.2. The circular shifts z^0, z^2 and z^9 used in this example are only for illustrative purposes. The output of stage 0, of the FFAST front-end, is then obtained by computing the 4-point DFT of each of the sub-sampled stream and further grouping them into 'bins'. A 3-dimensional vector $\vec{y}_{b,i,j}$ denotes the observation of bin j of stage i, e.g., $\vec{y}_{b,0,1}$ is the observation of bin 1 of stage 0.

bin-observation vectors and the DFT coefficients of the input signal $\vec{x}$, can be written as,

$$\begin{aligned}
\vec{y}_{b,0,0} &= \begin{pmatrix} w_{b,0,0}[0] \\ w_{b,0,0}[1] \\ w_{b,0,0}[2] \end{pmatrix}, \quad \vec{y}_{b,0,2} = \begin{pmatrix} 1 \\ e^{\imath 2\pi 20/20} \\ e^{\imath 2\pi 90/20} \end{pmatrix} X[10] + \begin{pmatrix} w_{b,0,2}[0] \\ w_{b,0,2}[1] \\ w_{b,0,2}[2] \end{pmatrix}, \\
\vec{y}_{b,0,1} &= \begin{pmatrix} 1 \\ e^{\imath 2\pi 2/20} \\ e^{\imath 2\pi 9/20} \end{pmatrix} X[1] + \begin{pmatrix} 1 \\ e^{\imath 2\pi 10/20} \\ e^{\imath 2\pi 45/20} \end{pmatrix} X[5] + \begin{pmatrix} 1 \\ e^{\imath 2\pi 26/20} \\ e^{\imath 2\pi 117/20} \end{pmatrix} X[13] + \begin{pmatrix} w_{b,0,1}[0] \\ w_{b,0,1}[1] \\ w_{b,0,1}[2] \end{pmatrix}
\end{aligned} \tag{4.1}$$

The FFAST algorithm proceeds as follows:

- *Divides the problem into simpler sub-problems*: The FFAST sub-sampling front-end takes a 20-length noise-corrupted observation $\vec{y}$, and disperses it into 4 bin observations, as shown in Fig. 4.1. Each bin has 3 output samples (corresponding to the 3 delay chains), and forms an instance of a sub-problem of computing a sparse DFT of a high-dimensional signal. Most sub-problems are trivial, consisting of computing a 0-sparse DFT, e.g., $\vec{y}_{b,0,0}$, or a 1-sparse DFT, e.g., $\vec{y}_{b,0,2}$, while the others are almost trivial, of computing a 3-sparse DFT, e.g., $\vec{y}_{b,0,1}$.

- *Iterative peeling-decoder*: The FFAST decoder identifies an instance of a sub-problem that is 1-sparse and reliably computes the support and the value of the non-zero DFT coefficient, e.g., $X[10]$ in $\vec{y}_{b,0,2}$, participating in this sub-problem. Then, it *peels off* the contribution of the identified non-zero DFT coefficient, from other sub-problems, to

create more instances of 1-sparse sub-problems. This peeling-style iterative recovery algorithm eventually uncovers all the non-zero DFT coefficients.

- *Noise robust processing*: The circular shift (or delay) pattern, used in the FFAST front-end architecture, determines the structure of the effective measurement matrix $\boldsymbol{A}_{i,j} \in \mathbb{C}^{3\times 20}$, of the sub-problem corresponding to bin j of stage i. The noise robust decoding of the 1-sparse sub-problems is achieved by judiciously choosing the delay pattern, such that the measurement matrices of all the sub-problems have good mutual incoherence property, i.e., the columns of the measurement matrix are uncorrelated, like Restricted-Isometry-Property (RIP) [12].

At a high level, the FFAST architecture through its multi-stage sub-sampling front-end, divides the original "difficult" (k-sparse) problem into many "simpler" (1-sparse) sub-problems. Then, it solves the 1-sparse sub-problems *reliably*, in the presence of observation noise, using multiple D measurements per sub-problem and iterates. Reliable decoding of 1-sparse sub-problems is achieved by 1) using carefully designed bin-measurement matrices $\boldsymbol{A}_{i,j}$ and 2) using a robust bin-processing/reconstruction algorithm.

The rest of the chapter is organized as follows: In Section 4.2, we provide the problem formulation along with the modeling assumptions of the signal and observation noise. Section 4.3 provides the main result of this chapter. In Section 4.4, we provide a brief overview of the related literature and contrast it with the results of this chapter. In Section 4.5, we exemplify the connection between the sampling pattern of the FFAST front-end and the resulting bin-measurement matrices. Section 4.6 describes the noise-robust version of the FFAST algorithm. In Section 4.7, we provide extensive simulations for various settings to empirically validate the performance of the FFAST algorithm in the presence of observation noise. Additionally, we also demonstrate an application of the FFAST framework for an MRI acquisition.

4.2 Signal model and Problem formulation

Consider an n-length discrete-time signal $\vec{x}$ that is a sum of $k << n$ complex exponentials, i.e., its n-length discrete Fourier transform has k non-zero coefficients:

$$x[p] = \sum_{q=0}^{k-1} X[\ell_q] e^{2\pi \imath \ell_q p/n}, \qquad p = 0, 1, \ldots, n-1, \tag{4.2}$$

where the discrete frequencies $\ell_q \in \{0, 1, \ldots, n-1\}$ and the amplitudes $X[\ell_q] \in \mathbb{C}$, for $q = 0, 1, \ldots, k-1$. We consider the problem of computing the k sparse n length DFT $\vec{X}$ when the observed time-domain samples are corrupted by white Gaussian noise, i.e., we can only observe a subset of the samples of $\vec{y}$, where,

$$\vec{y} = \vec{x} + \vec{w}, \tag{4.3}$$

and $\vec{w} \in \mathcal{C}N(0, I_{n\times n})$[3].

Further, we make the following modeling assumptions on the signal and the noise:

- All the non-zero DFT coefficients have magnitude $\sqrt{\rho}$, where ρ is signal-to-noise ratio, while the phase is chosen uniformly at random from the set $\{2\pi i/M\}_{i=0}^{M-1}$, for some fixed constant M (see Remark 4.3.2 for general constellation).
- The support of the non-zero DFT coefficients is uniformly random in the set $\{0, 1, \ldots, n-1\}$.
- The signal-to-noise ratio ρ, is defined as,

$$\rho = \min_{X[\ell]\neq 0} \frac{|X[\ell]|^2}{\mathbb{E}\{||\vec{w}||^2\}}.$$

Note, when $\vec{x}$ is 1-sparse, $\rho = \frac{||\vec{x}||^2}{\mathbb{E}\{||\vec{w}||^2\}}$.

4.3 Main results

We provide a robust version of the FFAST algorithm that processes the noise-corrupted time-domain observations, $\vec{y} = \vec{x} + \vec{w}$, to reliably compute the k-sparse DFT $\vec{X}$, i.e., $||\vec{X}||_0 \leq k$. The observation noise $\vec{w} \sim \mathcal{C}N(0, I_{n\times n})$ and the non-zero DFT coefficients of the signal $\vec{x}$ are assumed to have magnitude $\sqrt{\rho}$, where ρ denotes signal-to-noise-ratio, and phase uniformly random in the set $\{2\pi i/M\}_{i=0}^{M-1}$, for some constant M. A precise statement of the main result is given by the following theorem.

Theorem 4.3.1. *For any given $0 < \varepsilon < 1$ and a finite signal-to-noise-ratio, there exist (infinitely many) sufficiently large n, such that the FFAST algorithm computes the k-sparse DFT $\vec{X}$, where $k = \Omega(n^\delta)$ and $0 < \delta < 1$, of an n-length signal $\vec{x}$ from its noise-corrupted samples $\vec{y}$, with the following properties:*

- **Sample complexity**: *The algorithm needs $m = O(k \log^2(k) \log(n))$ samples of $\vec{y}$.*
- **Computational complexity**: *The computational complexity of the algorithm is $O(n \log(n) \log^2(k))$.*
- **Probability of success**: *The algorithm successfully recovers all the non-zero DFT coefficients of the signal $\vec{x}$, with probability at least $1 - \varepsilon$.*

Proof. Please see Appendix 4.C □

[3]The energy of signal is chosen appropriately to achieve the targeted signal-to-noise-ratio.

Remark 4.3.2. *[finite constellation] The FFAST framework and the reconstruction algorithm generalize to arbitrary complex-valued non-zero DFT coefficients, as long as the magnitudes of all the non-zero DFT coefficients are above signal-to-noise-ratio. The proof techniques for the arbitrary complex-valued non-zero DFT coefficients becomes much cumbersome and we do not address this issue in this chapter. At high level, the difficulty in the analysis arises due to the fact that when the non-zero DFT coefficients have arbitrary complex values, the FFAST decoders value-estimate is not perfect. Hence, the peeling-off step, of the iterative decoder, leaves behind a residual signal, i.e., self-interference, in the observation of other bins. The statistics of the interference plus noise is no more Gaussian, thus making the analysis technically cumbersome. Nonetheless, as shown in Chapter 3, the sparse bi-partite aliasing graph has a local tree-like neighborhood. Using this fact one can show that every singleton bin suffers at the most from $O(1)$ self-interference components and hence decoder succeeds with similar guarantees as in Theorem 4.3.1.*

4.4 Related work

The problem of computing a sparse discrete Fourier transform of a signal is related to the rich literature of frequency estimation [64, 62, 71, 69] in statistical signal processing as well as compressive-sensing [20, 8]. In frequency estimation, it is assumed that a signal consists of k complex exponentials in the presence of noise and the focus is on 'super-resolution' spectral estimation techniques based on well-studied statistical methods like MUSIC and ESPRIT [64, 62, 71, 69]. The methods used are based on subspace decomposition principles, e.g., singular-value-decomposition, which very quickly become computationally infeasible as the problem dimensions (k, n) increase. In contrast, we take a different approach combining tools from coding theory, number theory, graph theory and statistical signal processing, to divide the original problem into many instances of simpler problems. The divide-and-conquer approach of FFAST alleviates the scaling issues in a much more graceful manner.

In compressive sensing, the bulk of the literature concentrates on random linear measurements, followed by either convex programming or greedy pursuit reconstruction algorithms [8, 9, 75]. A standard tool used for the analysis of the reconstruction algorithms is the *restricted isometry property* (RIP) [12]. The RIP characterizes matrices which are nearly orthonormal or unitary, when operating on sparse vectors. Although random measurement matrices like Gaussian matrices exhibit the RIP with optimal scaling, they have limited use in practice, and are not applicable to our problem of computing a sparse DFT from time-domain samples. So far, to the best of our knowledge, the tightest characterization of the RIP, of a matrix consisting of random subset of rows of an $n \times n$ DFT matrix, provides a sub-optimal scaling, i.e., $O(k \log^3 k \log n)$, of samples [66]. The analysis of the FFAST recovery algorithm uses the RIP condition of individual bin-measurement matrices rather than the full measurement matrix, thus providing a better scaling for the number of samples, i.e., $O(k \log^2 k \log n)$. An alternative approach, in the context of sampling a continuous time

signal with a finite rate of innovation is explored in [76, 21, 5, 58].

At a higher level though, despite some key differences in our approach to the problem of computing a sparse DFT, our problem is indeed closely related to the spectral estimation and compressive sensing literature, and our approach is naturally inspired by this, and draws from the rich set of tools offered by this literature.

A number of previous works [31, 32, 35, 38, 39] have addressed the problem of computing a 1-D DFT of a discrete-time signal that has a sparse Fourier spectrum, in sub-linear sample and time complexity. Most of these algorithms achieve a sub-linear time performance by first isolating the non-zero DFT coefficients into different bins, using specific filters or windows that have 'good' (concentrated) support in both, time and frequency. The non-zero DFT coefficients are then recovered iteratively, one at a time. Although the time complexity of these algorithms sub-linear, their robustness to observation noise is limited. In [42], the author proposes a sub-linear time algorithm with a sample complexity of $O(k \log^4 n)$ or $O(k^2 \log^4 n)$ and computational complexity of $O(k \log^5 n)$ or $O(k^2 \log^4 n)$ to compute a sparse DFT, with high probability or zero-error respectively. The algorithm in [42] exploits the Chinese-Remainder-Theorem (CRT), along with $O(poly(\log n))$ number of subsampling patterns to identify the locations of the non-zero DFT coefficients. In contrast, the FFAST algorithm exploits the CRT to induce 'good' sparse-graph codes using a small constant number of subsampling patterns and computes the sparse DFT with a vanishing probability of failure.

4.5 FFAST sampling pattern and the measurement matrix

In this section, we describe a connection between the sampling pattern of the FFAST front-end and the measurement matrix of a sparse recovery problem. Consider again the example signal $\vec{x}$ of length $n = 20$, whose 20-point DFT $\vec{X}$ is $k = 5$ sparse. The 5 non-zero DFT coefficients of $\vec{x}$ are $X[1]$, $X[3]$, $X[5]$, $X[10]$ and $X[13]$. The noise-corrupted signal $\vec{y} = \vec{x} + \vec{w}$, is processed through stage 0 of a 2-stage FFAST sub-sampling front-end as shown in Fig. 4.1. The output samples of the FFAST front-end are grouped into 'bins', as shown in Fig. 4.1. Using the basic signal processing identities of sampling-aliasing and circular shifts, the relation between the bin-observation vectors and the DFT coefficients of the input signal $\vec{x}$, can be computed, e.g., see equation (4.1).

More generally, the observation vector $\vec{y}_{b,i,j}$ of bin j of stage i in a FFAST front-end architecture with d stages and D delays is given as,

$$\vec{y}_{b,i,j} \quad = \quad \boldsymbol{A}_{i,j}\vec{X} + \vec{w}_{b,i,j}, \quad 0 \leq i < d, \; 0 \leq j < D, \tag{4.4}$$

where $\vec{w}_{b,i,j} \sim \mathcal{C}N(0, I_{D\times D})$ and $\boldsymbol{A}_{i,j}$ is the *bin-measurement* matrix. Next, we describe the generic structure of bin-measurement matrices for different bins.

4.5.1 Bin-measurement matrix

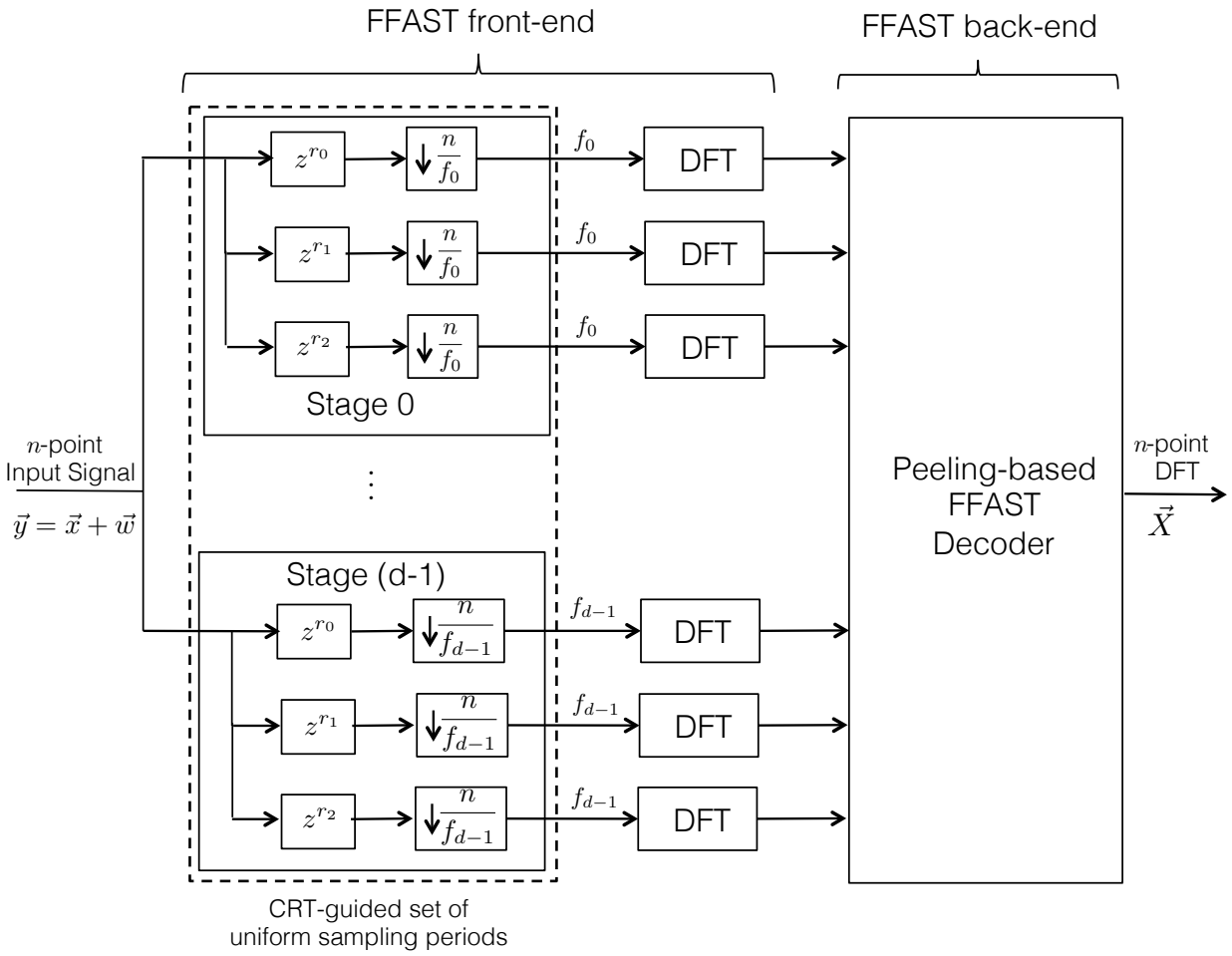


Figure 4.2: Schematic block diagram of the FFAST architecture for processing noise-corrupted samples $\vec{y} = \vec{x} + \vec{w}$. The n-point input signal $\vec{y}$ is uniformly subsampled by a carefully chosen set of d patterns, guided by the Chinese-Remainder-Theorem. Further, each stage has D circularly shifted and sampled streams of data, e.g., $D = 3$ in this figure. Each sub stream of sampled data in stage i has f_i number of samples. The delay shift pattern (r_0, r_1, r_2) is chosen carefully so as to induce bin-measurement matrices with "good" mutual incoherence property and the RIP (as explained in Section 4.5.1). Next, the (short) DFTs, of each of the sub-streams are computed using an efficient FFT algorithm of choice. The big n-point DFT $\vec{X}$ is then synthesized from the smaller DFTs using the peeling-like FFAST decoder.

Consider the FFAST front-end architecture that processes the noise-corrupted observations $\vec{y} = \vec{x} + \vec{w}$, with d stages and D delay sub-streams per stage as shown in Fig. 4.2. For the case when the observations are noiseless, $D = 2$ is sufficient to reconstruct the sparse DFT $\vec{X}$ from the sub-sampled data. However when the observed samples are corrupted by *noise*, $D > 2$ is necessary. As shown in Fig. 4.2, the i^{th} delay sub-stream of the j^{th} stage, circularly shifts the input signal by r_i and then sub-samples by a sampling period of n/f_j,

where f_j is the number of samples in each delay sub-stream of stage j. The number of delays D and the shift pattern $(r_0, \ldots, r_{D-1})$ influence bin-measurement matrices as shown below.

Let $\vec{a}(\ell)$, be a discrete D-dimensional complex vector given by:

$$\vec{a}(\ell) = \begin{pmatrix} e^{\imath 2\pi \ell r_0/n} \\ e^{\imath 2\pi \ell r_1/n} \\ \vdots \\ e^{\imath 2\pi \ell r_{D-1}/n} \end{pmatrix}, \tag{4.5}$$

in the sequel, we refer to $\vec{a}(\ell)$, $\ell = 0, 1 \ldots, n-1$, as a *steering-vector* of frequency $2\pi\ell/n$ sampled at $(r_0, \ldots, r_{D-1})$. Then, the bin-measurement matrix $\boldsymbol{A}_{i,j} \in \mathbb{C}^{D \times n}$ of bin j of stage i is given by:

$$\vec{\boldsymbol{A}}_{i,j}(\ell) = \begin{cases} \vec{a}(\ell) & \text{if } \ell \equiv j \bmod f_i \\ \vec{0} & \text{otherwise,} \end{cases} \tag{4.6}$$

where $\vec{\boldsymbol{A}}_{i,j}(\ell)$ is the ℓ^{th} column, $\ell = 0, \ldots, n-1$, of $\boldsymbol{A}_{i,j}$. For example in equation (4.1), the bin-measurement matrix $\boldsymbol{A}_{0,1}$ is,

$$\boldsymbol{A}_{0,1} = \left[\begin{array}{cccccccccccccccccccc} \vec{0} & \vec{a}(1) & \vec{0} & \vec{0} & \vec{0} & \vec{a}(5) & \vec{0} & \vec{0} & \vec{0} & \vec{a}(9) & \vec{0} & \vec{0} & \vec{0} & \vec{a}(13) & \vec{0} & \vec{0} & \vec{0} & \vec{a}(17) & \vec{0} & \vec{0} \end{array} \right],$$

where $\vec{a}(\ell) = \begin{pmatrix} 1 \\ e^{\imath 2\pi \ell/20} \end{pmatrix}$ for $\ell = 0, 1, \ldots, 19$.

Thus, the FFAST front-end effectively divides the problem of recovering a k-sparse DFT $\vec{X}$ of an n-length signal $\vec{x}$ from its noise-corrupted samples $\vec{y}$ into multiple bin-level problems of the form (4.4). The FFAST peeling-decoder then detects which of these bin-level problems are "single-tons", solves them (identifies the support and the value of the single-ton) using bin observation $\vec{y}_{b,i,j}$ and "peels-off" their contribution from other bin observations to create more singleton bins. In the case when the observations were not corrupted by noise, $D = 2$ measurements per bin were sufficient. However, in the presence of observation noise we need more measurements per bin, i.e., $D > 2$, to make bin-processing reliable and robust against noise. Moreover, the structure of the delays, i.e., the choice of the circular shifts $(r_0, \ldots, r_{D-1})$, also plays a crucial role in the design of the measurement matrices of the individual bins as shown in (4.6), which is further used to make the individual bin-processing robust against observation noise.

Incoherence properties of bin-measurement matrices In the compressed sensing literature the *mutual-incoherence* and the *Restricted-isometry-property* (RIP) [12], of the measurement matrix, are widely studied and well understood to play an important role in stably recovering a high-dimensional sparse vector from linear measurements in the presence of observation noise. The problem in (4.4) is a special case of the compressed sensing problem. In particular, if the processed bin is a single-ton, then the bin-processing algorithm attempts

to recover a 1-sparse high-dimensional signal from linear measurements in the presence of observation noise. We establish the *mutual-incoherence* and the RIP of bin-measurement matrices and use them to analyze the noise-robustness of the FFAST algorithm. Next, we define the mutual-incoherence property and the RIP for a general measurement matrix $\boldsymbol{A}$.

Definition 4.5.1. *The mutual incoherence $\mu_{\max}(\boldsymbol{A})$ of a measurement matrix $\boldsymbol{A}$ is defined as*

$$\mu_{\max}(\boldsymbol{A}) \triangleq \max_{\forall p \neq q} \frac{|\vec{\boldsymbol{A}}(p)^{\dagger}\vec{\boldsymbol{A}}(q)|}{||\vec{\boldsymbol{A}}(p)|| \cdot ||\vec{\boldsymbol{A}}(q)||}, \tag{4.7}$$

where $\vec{\boldsymbol{A}}(p)$ is the p^{th} column of the matrix $\boldsymbol{A}$.

The mutual-incoherence property of the measurement matrix indicates the level of correlation between the distinct columns of the measurement matrix. Smaller value of $\mu_{\max}(\boldsymbol{A})$ implies more stable recovery, e.g., $\mu_{\max}(\boldsymbol{A}) = 0$ for an orthogonal measurement matrix $\boldsymbol{A}$.

Definition 4.5.2. *The restricted-isometry constant $\gamma_s(\boldsymbol{A})$ of a measurement matrix $\boldsymbol{A}$, with unit norm columns, is defined to be the smallest positive number that satisfies*

$$(1 - \gamma_s(\boldsymbol{A}))||\vec{X}||^2 \leq ||\boldsymbol{A}\vec{X}||^2 \leq (1 + \gamma_s(\boldsymbol{A}))||\vec{X}||^2, \tag{4.8}$$

for all $\vec{X}$ such that $||\vec{X}||_0 \leq s$.

The RIP characterizes the norm-preserving capability of the measurement matrix when operating on sparse vectors. If the measurement matrix $\boldsymbol{A}$ has a good RIP constant (small value of $\gamma_{2s}(\boldsymbol{A})$) for all $2s$-sparse vectors, then a stable recovery can be performed for any s-sparse input vector [10]. This is because a small value of $\gamma_{2s}(\boldsymbol{A})$ implies that $||\boldsymbol{A}(\vec{X}_1 - \vec{X}_2)||^2$ is bounded away from zero for any two distinct, $\vec{X}_1 \neq \vec{X}_2$, s-sparse vectors.

Thus stable recovery performance of the FFAST algorithm is a function of the constants $\mu_{\max}$ and γ_s. From equation (4.6) it is clear that these constants dependent on the number of delays D as well as on the choice of the delay pattern $(r_0, \ldots, r_{D-1})$, i.e., sub-sampling patterns of the FFAST front-end. There are multiple options for choosing the delay patterns, e.g., uniform periodic delays, aperiodic deterministic delays, random delays, etc. We would like to choose a delay pattern that has good constants $\mu_{\max}$ and γ_s, to guarantee a stable recovery performance. Next, we describe the delay pattern used in the FFAST architecture and establish its mutual incoherence property and the RIP.

4.5.2 FFAST sampling patterns

A FFAST front-end architecture with d-stages has $\{f_i\}_{i=0}^{d-1}$ samples per sub-stream in each of the d stages (see Fig. 4.2). For a given (k, n), in Chapter 3, we show how to choose the parameters $\{f_i\}_{i=0}^{d-1}$ using the CRT. As an example consider a $d = 3$ stage FFAST front-end architecture such that the sampling period of stage $i = 0, 1, 2$, is $T_i = n/f_i$. Without loss

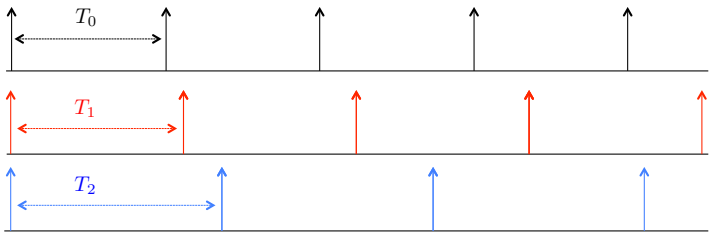


Figure 4.3: Principal sampling pattern, corresponding to the $r_0 = 0$ delay, of the FFAST front-end with $d = 3$ stages. The sampling periods of the three stages are T_0, T_1, T_2 respectively. The overall sampling pattern is a union of periodic sampling patterns.

of generality we assume that $r_0 = 0$. This results in a basic sampling pattern, as shown in Fig. 4.3, that we refer to as "principal sampling pattern" in the sequel. We use the principal sampling pattern in conjunction with the choice of the remaining $D-1$ delays $(r_1, \ldots, r_{D-1})$ to construct the overall FFAST sampling pattern. For example, sampling pattern of a $d = 3$ stages and $D = 3$ uniformly random delays, (r_0, r_1, r_2), is shown in Fig. 4.6.

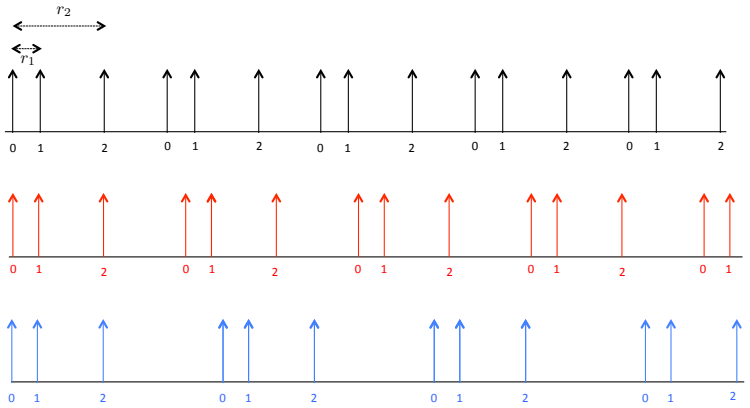


Figure 4.4: Sampling pattern of the FFAST front-end with $d = 3$ stages and $D = 3$ delay sub-stream per stage. The sampling periods of the three stages are T_0, T_1, T_2 respectively. Sampling impulse train corresponding to the delay sub stream 0 is indexed by 0 and so on. The delay pattern (r_0, r_1, r_2) is chosen uniformly at random from the set $\{0, 1 \ldots, n-1\}$. For sake of representation WLOG $r_0 = 0$.

Next, we characterize the mutual incoherence and the RIP properties of the bin measurement matrices, of the FFAST architecture, when delays are chosen uniformly at random from the set $\{0, \ldots, n-1\}$. In the sequel we use these properties to prove the stable recovery of the proposed FFAST peeling-style iterative recovery algorithm.

Lemma 4.5.3. *The mutual incoherence $\mu_{\max}(\boldsymbol{A}_{i,j})$ of the bin-measurement matrix $\boldsymbol{A}_{i,j}$, of the FFAST front-end with D delays, is upper bounded by*

$$\mu_{\max}(\boldsymbol{A}_{i,j}) < 2\sqrt{\log(5n)/D} \ , \ \forall \ i, j \tag{4.9}$$

with probability at least 0.2*, where the delays* $(r_0, \ldots, r_{D-1})$ *are chosen uniformly at random from the set* $\{0, 1, \ldots, n-1\}$.

Proof. Please see Appendix 4.A. □

Thus, for a random choice of delay pattern $r_0, \ldots, r_{D-1}$, the coherence parameter $\mu_{\max}(\boldsymbol{A}_{i,j})$ satisfies the bound in (4.9) with probability at least 0.2. Also, it is easy, i.e., $O(nD)$ complexity, to verify if a given choice of $r_0, \ldots, r_{D-1}$ satisfies (4.9) or not. Hence, using offline processing, one can choose a pattern $(r_0, \ldots, r_{D-1})$, such that deterministically the bound in (4.9) is satisfied.

Lemma 4.5.4. *The bin-measurement matrix* $\boldsymbol{A}_{i,j}$*, of the FFAST front-end with* D *delays, satisfies the following RIP condition for all* $\vec{X}$ *that have* $||\vec{X}||_0 \leq s$*,*

$$D\left(1-\mu_{\max}(\boldsymbol{A}_{i,j})(s-1)\right)^{+}||\vec{X}||^2 \leq ||\boldsymbol{A}_{i,j}\vec{X}||^2 \leq D\left(1+\mu_{\max}(\boldsymbol{A}_{i,j})(s-1)\right)||\vec{X}||^2 \ , \ \forall\ i,j \tag{4.10}$$

where the delays $(r_0, \ldots, r_{D-1})$ *are chosen uniformly at random from the set* $\{0, 1, \ldots, n-1\}$.

Proof. Please see Appendix 4.B □

4.6 Noise robust FFAST algorithm

A generic structure of the FFAST architecture, that processes noise-corrupted observations $\vec{y} = \vec{x} + \vec{w}$, is shown in Fig. 4.2. In Chapter 3, we described in detail how to carefully design the sub-sampling parameters, i.e., number of stages d and samples per sub-stream $\{f_i\}_{i=0}^{d-1}$ in each of the d stages, of the FFAST front-end architecture, using the CRT, for a given sparsity-index δ. For the case when there is no noise in the observations, we showed that $D = 2$ is sufficient to reconstruct the sparse DFT $\vec{X}$ from the sub-sampled data. However when the observed samples are corrupted by *noise*, as explained in Section 4.5, we use $D > 2$ number of delay sub-streams per stage. As shown in Fig. 4.2, the i^{th} delay sub-stream of the j^{th} stage, circularly shifts the input signal by r_i and then sub-samples by a sampling period of n/f_j, where f_j is the number of samples in each delay sub-stream of stage j. The number of delays D and the shift pattern $(r_0, \ldots, r_{D-1})$ influence the bin-measurement matrices as shown in Section 4.5.1. We choose the delay pattern $(r_0, r_1, \ldots, r_{D-1})$ such that bin-measurement matrices have "good" mutual incoherence property and the RIP.

From an algorithmic perspective, the only difference between the presence and the absence of observation noise is in the thresholds used by the algorithm and the subroutine 'Singleton-Estimator' that processes individual bins to test if it is a single-ton. In Algorithm 5 and Algorithm 6, we provide the pseudocode of the noise-robust FFAST algorithm.

Algorithm 5 FFAST Algorithm

Input: The noise-corrupted bin observations $\vec{y}_{b,i,j}$ obtained through the FFAST front-end of Fig. 4.2, for each bin j in stage i for all i, j.

Output: An estimate $\vec{X}$ of the k-sparse n-point DFT.

FFAST Decoding: Set the initial estimate of the n-point DFT $\vec{X} = 0$. Let ℓ denote the number of iterations performed by the FFAST decoder.
Set the energy threshold $T = (1 + \gamma)D$ for appropriately chosen γ (see Appendix 4.C).
for each iteration **do**
  for each stage i **do**
    for each bin j **do**
      if $||\vec{y}_{b,i,j}||^2 < T$ **then**
        bin is a *zero-ton*.
      else
        (singleton, v_p, p) = ***Singleton-Estimator*** $(\vec{y}_{b,i,j})$.
        if singleton = 'true' **then**
          Peel-off: $\vec{y}_{b,s,q} = \vec{y}_{b,s,q} - v_p\vec{a}(p)$, for all stages s and bins $q \equiv p \bmod f_q$.
          Set, $X[p] = v_p$.
        else
          bin is a *multi-ton*.
        end if
      end if
    end for
  end for
end for

4.7 Simulations

The FFAST front-end uses *coding-theoretic tools*, in particular CRT-guided sampling patterns that induce LDPC-like graphs in the frequency-domain, to divide the original problem of computing a k-sparse DFT $\vec{X}$, into multiple simpler (mostly 1-sparse) *bin level* compressed sensing problems. Next, we use techniques from *signal processing* and *estimation theory* to robustly solve these individual bin-level problems, in conjunction with iterative peeling. This modular approach of solving the problem, is exploited in providing a proof of Theorem 4.3.1

Algorithm 6 Singleton-Estimator

Inputs: The noise-corrupted bin observation $\vec{y}_{b,i,j}$ and indices (i, j) indicating the stage and the bin number respectively.

Outputs: 1) A boolean flag 'singleton', 2) Estimate of the value v_p of the non-zero DFT coefficient and 3) the position p of the non-zero DFT coefficient.

Singleton-Estimator:
Set the singleton = 'false'.
Set the energy threshold $T = (1 + \gamma)D$ for appropriately chosen γ (see Appendix 4.C).
for each position $q \equiv i \bmod f_j$ for all $q = 0, \ldots, n-1$ **do**
  $v_q = \vec{a}(q)^\dagger \vec{y}_{b,i,j}/D$.
  if $||\vec{y}_{b,i,j} - v_q \vec{a}(q)||^2 < T$ **then**
    singleton = 'true'.
    $p = q$ and $v_p = v_q$.
  end if
end for

in Appendix 4.C. The analytical result of Theorem 4.3.1, can be interpreted as,

$$\begin{aligned} \#\text{ of samples } m \text{ required by FFAST} &= \{\#\text{ of bins required for successful decoding} \\ &\quad \text{of the resulting sparse graph}\} \times \\ &\quad \{\#\text{ of delays required for robust bin-processing}\} \\ &= \{c_1(\delta)k\} \times \left\{c_2(\rho)c_3 \log^2 k \log n\right\} \end{aligned} \tag{4.11}$$

where the constant $c_1(\delta)$ depends on the sparsity index $0 < \delta < 1$ and constant $c_2(\rho)$ is a function of signal-to-noise-ratio ρ. In coding theory, it is now well understood that the optimal scaling of the number of bins, required for successful decoding of a sparse graph code with k variable nodes, is $O(k)$. On contrary, the scaling $O(\log^2 k \log n)$ for the number of delays, required for robust bin-processing, is sub-optimal and is an artifact of our analytical techniques. We believe that the actual performance of the FFAST algorithm is,

$$\begin{aligned} \#\text{ of samples } m \text{ required by FFAST} &= \{\#\text{ of bins required for successful decoding} \\ &\quad \text{of the resulting sparse graph}\} \times \\ &\quad \{\#\text{ of delays required for robust bin-processing}\} \\ &= \{c_1(\delta)k\} \times \{c_2(\rho)c_3 \log(n)\}\,. \end{aligned} \tag{4.12}$$

The theoretical analysis of Theorem 4.3.1, falls short of proving this conjectured FFAST performance, due to technical difficulties. In this section we empirically explore the FFAST performance for various settings, in an attempt to validate the conjectured performance of

(4.12). In Chapter 3 we have shown theoretically and validated empirically that $c_1(\delta) < 2,\ \forall\ 0 < \delta < 0.99$.

4.7.1 Sample complexity m as a function of n

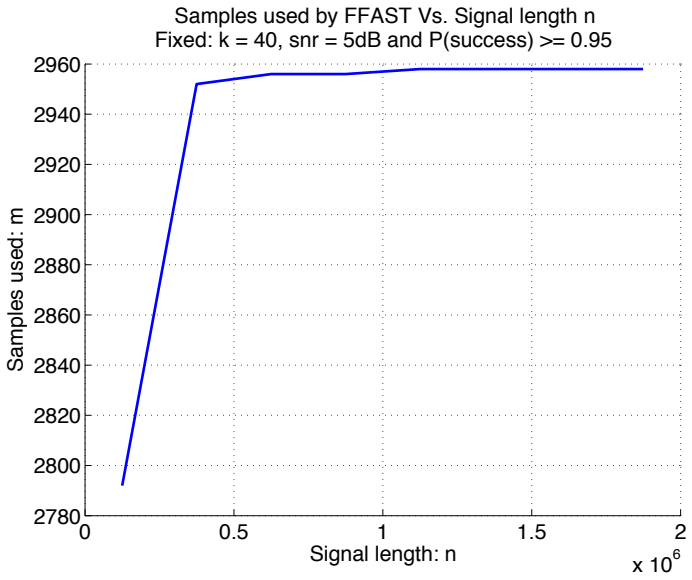


Figure 4.5: The plot shows the scaling of the number of samples m required by the FFAST algorithm to reliably reconstruct a $k = 40$ sparse DFT $\vec{X}$, from noise-corrupted observations $\vec{y} = \vec{x} + \vec{w}$, for increasing signal length n. For a fixed reliability, signal-to-noise ratio and sparsity, we note that m scales logarithmically with increasing n. This is consistent with the conjectured performance in (4.12).

In this section we empirically validate the scaling of the number of measurements m required by the FFAST algorithm to reliably compute the DFT $\vec{X}$ for various signal lengths n.

Simulation Setup

- An n-length DFT vector $\vec{X}$ with $k = 40$ non-zero coefficients is generated. The non-zero DFT coefficients are chosen to have uniformly random support from the set $\{0, \ldots, n-1\}$, with values from the set $\{\pm\sqrt{\rho}\}$[4] randomly. The input signal $\vec{x}$ is obtained by taking IDFT of $\vec{X}$. The length of the signal n is varied from $n = 49 * 50 * 51 \approx 0.1$

[4]We have also simulated the case where the amplitudes of the non-zero DFT coefficients are arbitrary complex numbers, with fixed magnitude and uniformly random phase, and obtained similar performance plots. In this section, we provide the simulation results for the case when the non-zero DFT coefficients take antipodal values, only to be consistent with the theoretical analysis of the chapter.

million to $n = 49 * 50 * 51 * 15 \approx 1.87$ million. This corresponds to the very-sparse regime of $k \propto n^{1/3}$. Note that the choice of the ambient dimension n, to be a product of approximately equal sized relatively co-prime numbers, is induced by the Chinese-Remainder-Theorem.

- A noise-corrupted signal $\vec{y} = \vec{x} + \vec{w}$, is generated by adding zero-mean, unit variance complex white Gaussian noise $\vec{w}$ with ρ chosen to have a signal-to-noise ratio of 5dB.
- The noise-corrupted signal $\vec{y}$ is input to a $d = 3$ stage FFAST architecture with D sub-streams per stage (see Fig. 4.2). The sampling periods of the 3 stages in the FFAST architecture are $50 * 51$, $51 * 49$ and $49 * 50$ respectively. This results in the number of samples per sub-stream, for the three stages to be $f_0 = 49$, $f_1 = 50$ and $f_2 = 51$ respectively. Thus the total number of bins in the FFAST front-end architecture is $n_b = \sum_{i=0}^{2} f_i = 150 = 3.75k$, i.e., $c_1(\delta) = 3.75$.
- As the signal length n varies the number of delays D per bin are varied to have reliable decoding, i.e., $Pr(\text{success}) \geq 0.95$. For each value of D, the total number of samples used by the FFAST are $m = n_b * D = 3.75 * k * D$.
- Each sample point in the plot is generated by averaging the performance of FFAST over 1000 runs for each configuration.

We note that the number of samples, in particular D, increase *logarithmically* with increasing n. Thus, empirically confirming the conjectured performance of $m = (3.75k) * (c_2(\rho)c_3 \log(n))$.

4.7.2 Sample complexity m as a function of ρ

In this section we experimentally probe the scaling of the number of measurements m as a function of signal-to-noise-ratio ρ, i.e., $c_2(\rho)$.

Simulation Setup

- An $n = 49 * 50 * 51 \approx 0.1$ million length, DFT vector $\vec{X}$, with $k = 40$ non-zero coefficients is generated. The non-zero DFT coefficients are chosen to have uniformly random support from the set $\{0, \ldots, n-1\}$, and the values from the set $\{\pm\sqrt{\rho}\}$. The input signal $\vec{x}$ is obtained by taking IDFT of $\vec{X}$.
- A noise-corrupted signal $\vec{y} = \vec{x} + \vec{w}$, is generated by adding zero-mean, unit variance complex white Gaussian noise $\vec{w}$.
- The parameter ρ is varied such that the effective signal-to-noise ratio ranges from -1dB to 9dB.

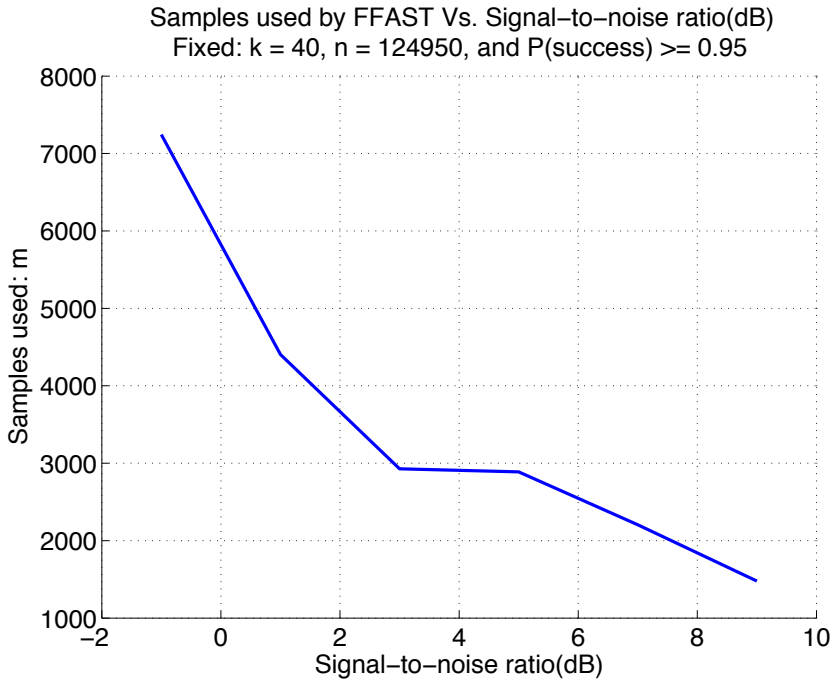


Figure 4.6: The plot shows the scaling of the number of samples m required by the FFAST algorithm to reliably compute an $n \approx 0.1$ million, $k = 40$ sparse DFT $\vec{X}$, from noise-corrupted observations $\vec{y} = \vec{x} + \vec{w}$, for increasing signal-to-noise ratio ρ. For fixed values of all other parameters, the number of samples decreases roughly in an *inverse relation* with the increasing signal-to-noise ratio on log-scale, i.e., in (4.12) the parameter $c_2(\rho) \propto 1/\log(\rho)$.

- The noise-corrupted signal $\vec{y}$ is input to a $d = 3$ stage FFAST architecture with D sub-streams per stage (see Fig. 4.2). The sampling periods of the 3 stages in the FFAST architecture are $50 * 51$, $51 * 49$ and $49 * 50$ respectively. This results in the number of samples per sub-stream, for the three stages to be $f_0 = 49$, $f_1 = 50$ and $f_2 = 51$ respectively. Thus the total number of bins in the FFAST front-end architecture is $n_b = \sum_{i=0}^{2} f_i = 150 = 3.75k$, i.e., $c_1(\delta) = 3.75$.

- As the signal-to-noise ratio varies the number of delays D per bin are varied to have reliable decoding, e.g., $Pr(\text{success}) \geq 0.95$.

- Each sample point in the plot is generated by averaging the performance of FFAST over 1000 runs for each configuration.

The number of samples decrease roughly in an *inverse relation* with increasing signal-to-noise ratio on log-scale, i.e., in (4.12) the parameter $c_2(\rho) \propto 1/\log(\rho)$.

4.A Mutual incoherence bound

In this section we provide a proof of Lemma 4.5.3. Let $\vec{a}(p)$ be a D-dimensional steering vector with frequency $2\pi p/n$, for $p = 0, \ldots, n-1$, as given in equation (4.5). Then,

$$\begin{aligned} \mu_{\max}(\boldsymbol{A}_{i,j}) &\leq \max_{p \neq q} \frac{1}{D} |\vec{a}(p)^{\dagger} \vec{a}(q)| \\ &= \max_{\ell \neq 0} \frac{1}{D} |\sum_{s=0}^{D-1} \exp(\imath 2\pi \ell r_s/n)| \\ &= \max_{\ell \neq 0} \mu(\ell) \ , \end{aligned} \tag{4.13}$$

where $\mu(\ell) \triangleq |\sum_{s=0}^{D-1} \exp(\imath 2\pi \ell r_s/n)|/D$.

Now consider the summation $\sum_{s=0}^{D-1} \cos(\imath 2\pi \ell r_s/n)/D$ for any fixed $\ell \neq 0$. Each term in the summation is a zero-mean random variable i.i.d with bounded support in $[-1/D, 1/D]$. Thus, using Hoeffding's inequality for the sum of independent random variables with bounded support we have, for any $t > 0$,

$$Pr(|\sum_{s=0}^{D-1} \cos(\imath 2\pi \ell r_s/n)/D| > t) \leq 2\exp(-t^2 D/2).$$

Similarly,

$$Pr(|\sum_{s=0}^{D-1} \sin(\imath 2\pi \ell r_s/n)/D| > t) \leq 2\exp(-t^2 D/2).$$

Applying a union bound over the real and the imaginary parts of the summation term in $\mu(\ell)$, we get,

$$Pr(\mu(\ell) > \sqrt{2}t) \leq 4\exp(-t^2 D/2). \tag{4.14}$$

Further applying a union bound over all $\ell \neq 0$, we have,

$$\begin{aligned} Pr(\mu_{\max}(\boldsymbol{A}_{i,j}) > \sqrt{2}t) &\leq 4n\exp(-t^2 D/2) \\ &= 0.8 \end{aligned}$$

for $t = \sqrt{2\log(5n)/D}$. Thus, over all the random choices of the delays $r_0, \ldots, r_{D-1}$, with probability at least 0.2,

$$\mu_{\max}(\boldsymbol{A}_{i,j}) < 2\sqrt{\log(5n)/D} \ , \quad \forall \ i, j.$$

■

4.B Restricted-isometry-property

Consider a s-sparse vector $\vec{X}$ and a measurement matrix $\boldsymbol{A}_{i,j}$ corresponding to bin j of stage i of the FFAST. Using basic linear algebra we get the following inequality,

$$\lambda_{\min}(\boldsymbol{A}_{i,j}^{\dagger}\boldsymbol{A}_{i,j})||\vec{X}||^2 \leq ||\boldsymbol{A}_{i,j}\vec{X}||^2 \leq \lambda_{\max}(\boldsymbol{A}_{i,j}^{\dagger}\boldsymbol{A}_{i,j})||\vec{X}||^2$$

The Gershgorin circle theorem [29], provides a bound on the eigen-values of a square matrix. It states that, every eigen-value of a square matrix lies within at least one of the Gershgoring discs. A Gershgorin disc is defined for each row of the square matrix, with diagonal entry as a center and the sum of the absolute values of the off-diagonal entries as radius. Note that for all (i,j), irrespective of the values of the delays $r_0,\ldots,r_D$, the diagonal entries of the matrix $\boldsymbol{A}_{i,j}^{\dagger}\boldsymbol{A}_{i,j}$ are equal to D. Hence, $D\mu_{\max}(\boldsymbol{A}_{i,j})$ provides an upper bound on the absolute values of the off-diagonal entries of $\boldsymbol{A}_{i,j}^{\dagger}\boldsymbol{A}_{i,j}$.

Then, using the Gershgorin circle theorem we have,

$$D\left(1-\mu_{\max}(\boldsymbol{A}_{i,j})(s-1)\right)_{+}||\vec{X}||^2 \leq ||\boldsymbol{A}_{i,j}\vec{X}||^2 \leq D\left(1+\mu_{\max}(\boldsymbol{A}_{i,j})(s-1)\right)||\vec{X}||^2.$$

■

4.C Proof of Theorem 4.3.1

In this section we provide a proof of the main result of this chapter. The proof consists of two parts. In the first part of the proof, we show that the FFAST reconstructs the DFT $\vec{X}$, with probability at least $1-\varepsilon$ for any $\varepsilon>0$, using $m=O(k\log^2(k)\log(n))$ samples. In the second part of the proof we show that computational complexity of the FFAST decoder is $O(n\log^2(k)\log(n))$.

4.C.1 Reliability Analysis and sample complexity of the FFAST

Sample complexity

In Chapter 3, we have shown that for the noiseless case, for any given $0<\delta<1$ and sufficiently large (k,n), the FFAST algorithm computes the k-sparse n-length DFT $\vec{X}$, with probability at least $1-O(1/k)$, using a total of $O(k)$ number of bins. Later in Section 4.C.1 we show that $D=O(\log^2(k)\log(n))$ number of samples per bin are sufficient to make Algorithm 6 robust against observation noise. Hence, the total sample complexity of the FFAST algorithm in the presence of observation noise is

$$m=O(k\log^2(k)\log(n)).$$

Reliability analysis

In Lemma 4.5.3, we have shown that a random choice of delay parameters $r_0, \ldots, r_{D-1}$ satisfies the upper bound of (4.9) with probability at least 0.2. Also, it is easy, i.e., $O(nD)$ complexity, to verify if for a given choice of $r_0, \ldots, r_{D-1}$ (4.9) is satisfied or not. Hence WLOG henceforth we assume that the $r_0, \ldots, r_{D-1}$ are chosen such that deterministically the bound in (4.9) is satisfied.

Let E_b be an event that *a bin* processed by the Algorithm 6 is decoded wrongly. We first show that the Algorithm 6 processes each bin reliably, i.e., $Pr(E_b) < O(1/k^2)$, using $D = O(\log^2(k)\log(n))$ number of samples per bin. Then, we show that an event E that *some bin* is wrongly decoded by the FFAST algorithm, while reconstructing the DFT $\vec{X}$ has a low probability. In particular, we use a union bound over the constant number of iterations required for the FFAST to reconstruct the DFT $\vec{X}$ and over $O(k)$ bins used in the FFAST architecture to get,

$$\begin{aligned} Pr(E) &< \text{number of iterations} \times \text{number of bins} \times Pr(E_b) \\ &< O(1/k). \end{aligned} \tag{4.15}$$

Let E_f denote an event that the FFAST algorithm fails to reconstruct the DFT $\vec{X}$. Then putting the pieces together, we get,

$$\begin{aligned} Pr(E_f) &< Pr(E) + Pr(E_f \mid \overline{E}) \\ &\overset{(a)}{<} O(1/k) + O(1/k) \\ &< \varepsilon \end{aligned} \tag{4.16}$$

where in (a) we used the bound from (4.15) and the results from Chapter 3 along with the fact that if there is no error in any bin-processing, the FFAST algorithm performs as if it has access to the noiseless observations.

Hence, in order to complete the reliability analysis of the FFAST algorithm we need to show that $D = O(\log^2(k)\log(n))$ samples per bin are sufficient to achieve $Pr(E_b) < O(1/k^2)$. The following lemma, that analyzes the performance of an energy-based threshold rule to detect the presence of a complex vector in the presence of noise, plays a crucial role in the analysis of the event E_b.

Lemma 4.C.1. *For a complex vector $\vec{u} \in \mathbb{C}^D$ and $\vec{w} \sim \mathcal{C}N(0, I_{D\times D})$, we have,*

$$Pr(||\vec{u} + \vec{w}||^2 < (1+\gamma)D) < 2e^{-D\gamma^2/9} + e^{-(||\vec{u}|| - \sqrt{2\gamma D})_+^2}, \tag{4.17}$$

for any constant $0 < \gamma < 1$ and $D > 1.5/\gamma$.

Proof. Please see Appendix 4.D □

Without loss of generality consider processing of bin j from stage i of the FFAST architecture. As shown in the Section 4.5 bin observation noise is $\mathcal{C}N(0, I_{D\times D})$. Bin j in stage i is either a zero-ton bin, or a single-ton bin or a multi-ton bin. We analyze all the three events below and show that irrespective of the type, the Algorithm 6 decodes bin successfully with probability at least $1 - O(1/k^2)$, as long as $D = O(\log^2(k)\log(n))$.

Analysis of a zero-ton bin Consider a zero-ton bin with an observation $\vec{y}_{b,i,j} = \vec{w}_{b,i,j}$. Let E_z be an event that a zero-ton bin is not identified as a 'zero-ton'. Then,

$$\begin{aligned} Pr(E_z) &= Pr(||\vec{w}_{b,i,j}||^2 > (1+\gamma)D) \\ &= P(\chi^2_{2D} > 2(1+\gamma)D) \\ &< 2\exp(-D\gamma^2/9) \quad \forall \quad \gamma \in [0, 1/3]. \end{aligned} \tag{4.18}$$

where the last inequality follows from a standard concentration bound for Lipschitz functions of Gaussian variables, along with the fact that the Euclidean norm is a 1-Lipschitz function. Thus, $Pr(E_z) < O(1/k^2)$ if,

$$D > 18\log(k)/\gamma^2 \tag{4.19}$$

Analysis of a single-ton bin Let $\vec{y}_{b,i,j} = X[\ell]\vec{a}(\ell) + \vec{w}_{b,i,j}$, be an observation vector of a single-ton bin. The steering vector $\vec{a}(\ell)$ is the ℓ^{th} column of the bin-measurement matrix $\boldsymbol{A}_{i,j}$ and $X[\ell]$ is the only non-zero DFT coefficient connected to this bin. Let E_s be an event that a single-ton bin is not decoded correctly. The event E_s consists of the following three events.

Single-ton bin is wrongly classified as a zero-ton bin $[E_{sz}]$ Let E_{sz} denote an event that the single-ton bin fails the energy test of the Algorithm 6 and is classified as a zero-ton.

$$\begin{aligned} Pr(E_{sz}) &= Pr(||\vec{y}_{b,i,j}||^2 < (1+\gamma)D) \\ &= Pr(||X[\ell]\vec{a}(\ell) + \vec{w}_{b,i,j}||^2 < (1+\gamma)D) \\ &< 2\exp\{-D\gamma^2/9\} + \exp\{-(||X[\ell]\vec{a}(\ell)|| - \sqrt{2\gamma D})_+^2\} \end{aligned}$$

where the last inequality follows from Lemma 4.C.1. The non-zero DFT coefficient $X[\ell] = \sqrt{\rho}e^{\imath\phi}$ and the steering vector $||\vec{a}(\ell)|| = \sqrt{D}$. Hence,

$$Pr(E_{sz}) < 2\exp\{-D\gamma^2/9\} + \exp\{-(\sqrt{\rho D} - \sqrt{2\gamma D})_+^2\}. \tag{4.20}$$

Single-ton bin is wrongly classified as some other single-ton bin $[E_{ss}]$ Let E_{ss} denote an event that the Algorithm 6 wrongly concludes that the observation $\vec{y}_{b,i,j}$ corresponds

to a single-ton bin with steering vector $\vec{a}(\ell')$ and the non-zero DFT coefficient $X[\ell']$, for some $\ell' \neq \ell$. Then,

$$\begin{aligned} Pr(E_{ss}) &= Pr(||\vec{y}_{b,i,j} - X[\ell']\vec{a}(\ell')||^2 < (1+\gamma)D) \\ &= Pr(||X[\ell]\vec{a}(\ell) - X[\ell']\vec{a}(\ell') + \vec{w}_{b,i,j}||^2 < (1+\gamma)D) \\ &\overset{(a)}{=} Pr(||\boldsymbol{A}_{i,j}\vec{v} + \vec{w}_{b,i,j}||^2 < (1+\gamma)D) \\ &< 2\exp\{-D\gamma^2/9\} + \exp\{-(||\boldsymbol{A}_{i,j}\vec{v}|| - \sqrt{2\gamma D})_+^2\} \end{aligned} \tag{4.21}$$

where $\vec{v}$ used in (a) is an n-dimensional complex vector with only two non-zero values $v[\ell] = X[\ell]$ and $v[\ell'] = X[\ell']$, i.e., 2-sparse. The last inequality again follows from Lemma 4.C.1.

Using Lemma 4.5.3 and Lemma 4.5.4,

$$\begin{aligned} ||\boldsymbol{A}_{i,j}\vec{v}||^2 &\geq 2||\vec{v}||^2 D(1 - \mu_{\max}(\boldsymbol{A}_{i,j}))_+ \\ &= 2\rho D(1 - \mu_{\max}(\boldsymbol{A}_{i,j}))_+ \\ &\geq 2\rho D(1 - 2\sqrt{\log(5n)/D})_+. \end{aligned}$$

Thus, bound in (4.21) becomes,

$$Pr(E_{ss}) < 2\exp\{-D\gamma^2/9\} + \exp\left\{-\left(\sqrt{2\rho D(1 - 2\sqrt{\log(5n)/D})_+} - \sqrt{2\gamma D}\right)_+^2\right\} \tag{4.22}$$

Single-ton bin is wrongly classified as a multi-ton bin $[E_{sm}]$ Let E_{sm} be an event that bin is processed by the Algorithm 6 but no single-ton is found. Thus,

$$\begin{aligned} Pr(E_{sm}) &< Pr(E_{sm} \mid \hat{X[\ell]} = X[\ell]) + Pr(\hat{X[\ell]} \neq X[\ell]) \\ &= Pr(E_z) + Pr(\hat{X[\ell]} \neq X[\ell]) \end{aligned} \tag{4.23}$$

From Algorithm 6 we have $\hat{X[\ell]} = \vec{a}(\ell)^\dagger \vec{y}_{b,i,j}/D = X[\ell] + \mathcal{C}N(0, 1/D)$. Then, using the fact that non-zero DFT coefficients take value from a M-PSK constellation with magnitude $\sqrt{\rho}$ we have,

$$\begin{aligned} Pr(\hat{X[\ell]} \neq X[\ell]) &< Pr(|\mathcal{C}N(0, 1/D)| > \sqrt{\rho}\sin(\pi/M)) \\ &= \exp\{-D\rho\sin^2(\pi/M)\}. \end{aligned}$$

Substituting the above bound and (4.18) in (4.23), we get,

$$Pr(E_{sm}) < 2\exp\{-D\gamma^2/9\} + \exp\{-D\rho\sin^2(\pi/M)\}. \tag{4.24}$$

Further using a union bound, we get an upper bound on the probability of event E_s as,

$$Pr(E_s) < Pr(E_{sz}) + Pr(E_{ss}) + Pr(E_{sm})$$

Thus, from (4.20), (4.22) and (4.24), to reliably decode a single-ton bin with probability at least $1 - O(1/k^2)$ we need

$$D > \max\left\{\frac{2\log(k)}{(\sqrt{\rho}-\sqrt{2\gamma})_+^2}, 16\log(n), \frac{18\log(k)}{\gamma^2}, \frac{2\log(k)}{\rho\sin^2(\pi/M)}\right\} \tag{4.25}$$

Analysis of a multi-ton bin Consider a multi-ton bin, in particular a L-ton bin where $L \geq 2$. Then, the observation vector of this bin can be written as $\vec{y}_{b,i,j} = \boldsymbol{A}_{i,j}\vec{v} + \vec{w}_{b,i,j}$, where $\vec{v} \in \mathbb{C}^n$ is some L-sparse vector. Let E_m be an event that a multi-ton bin is decoded as a single-ton bin with a steering vector $\vec{a}(\ell)$ and the non-zero DFT coefficient $X[\ell]$ for some ℓ, i.e., $E_m = (||\boldsymbol{A}_{i,j}\vec{v} - X[\ell]\vec{a}(\ell) + \vec{w}_{b,i,j}||^2 < (1+\gamma)D)$.

First we compute a lower bound on the term $||\boldsymbol{A}_{i,j}\vec{v} - X[\ell]\vec{a}(\ell)||^2$ using the RIP Lemma 4.5.4. Let $\vec{u} = \vec{v} - X[\ell]\vec{e}_\ell$, where $\vec{e}_\ell$ is a standard basis vector with 1 at ℓ^{th} location and 0 elsewhere. The vector $\vec{u}$ is either $L+1$ or L sparse. Then, using the Lemmas 4.5.3, 4.5.4 and the fact that all the non-zero components of $\vec{u}$ are of the form $\sqrt{\rho}e^{\imath\phi}$, we have,

$$\begin{aligned} ||\boldsymbol{A}_{i,j}\vec{v} - X[\ell]\vec{a}(\ell)||^2 &= ||\boldsymbol{A}_{i,j}\vec{u}||^2 \\ &\geq L\rho D(1 - \mu_{\max}(\boldsymbol{A}_{i,j})L)_+ \\ &> L\rho D(1 - 2L\sqrt{\log(5n)/D})_+ \end{aligned} \tag{4.26}$$

Next, we compute an upper bound on the probability of the failure event E_m,

$$\begin{aligned} Pr(E_m) &< Pr(E_m \mid L < \log(k)) + Pr(L \geq \log(k)) \\ &< Pr(E_m \mid L < \log(k)) + O(1/k^2), \end{aligned} \tag{4.27}$$

where in last inequality we have used the fact that the number of the non-zero DFT coefficients connected to any bin L is a Binomial $B(1/(\eta k), k)$ random variable (see Chapter 3 for more details), for some constant $\eta > 0$. Hence to show that $Pr(E_m) < O(1/k^2)$, we need to show $Pr(E_m \mid L < \log(k)) < O(1/k^2)$. Towards that end,

$$\begin{aligned} Pr(E_m \mid L < \log(k)) &= Pr(||\boldsymbol{A}_{i,j}\vec{v} - X[\ell]\vec{a}(\ell) + \vec{w}_{b,i,j}||^2 < (1+\gamma)D \mid L < \log(k)) \\ &\overset{(a)}{<} \max_{2 \leq L < \log k} \exp\left\{-\left(\sqrt{L\rho D(1 - 2L\sqrt{\log(5n)/D})_+} - \sqrt{\gamma D}\right)_+^2\right\}, \end{aligned}$$

where in (a) we used the Lemma 4.C.1 and the lower bound from (4.26).

Hence, $Pr(E_m) < O(1/k^2)$, if

$$D > 4(1+\beta)\log^2(k)\log(n), \tag{4.28}$$

for some constant $\beta > 0$.

Upper bound on the probability of event E_b We set the threshold $\gamma = \min\{1, \rho/4\}$. Then, using (4.19), (4.25) and (4.28) we have,

$$Pr(E_b) < O(1/k^2), \tag{4.29}$$

for any fixed $\rho > 0$ and $D = O(\log^2(k)\log(n))$.

4.C.2 Computational complexity of the FFAST algorithm

The computational cost of the FFAST algorithm can be roughly computed as,

$$\begin{aligned}
\text{Total \# of operations} &= \text{\# of operations for the FFAST front-end} \\
&+ \text{\# of operations for the backend peeling-decoder} \\
&= O(k \log^3 k \log n) \\
&+ \text{\# of iterations} \times \{\text{\# of bins} \times (\text{operations/bin})\}.
\end{aligned}$$

As shown in Chapter 3 for all values of sparsity index $0 < \delta < 1$ the FFAST front-end employs no more than $O(k)$ number of bins and if successful completes decoding in constant number of iterations. Now, from the pseudocode of function Singleton-Estimator provided in Algorithm 6, it is clear that per bin, the FFAST decoding algorithm performs exhaustive search over $O(n/k)$ columns of the bin-measurement matrix, where each column is of dimension D. Further as shown in Section 4.C.1, the number of delays $D = O(\log^2(k)\log(n))$ is sufficient for reliable reconstruction. Thus, the overall computational complexity of the FFAST algorithm is no more than,

$$\begin{aligned}
\text{Total \# of arithmetic operations} &= O(k \log^3 k \log n) \\
&+ \text{\# of iterations} \times \{O(k) \times (O(n/k) \times D)\}, \\
&= O(k \log^3 k \log n) + O(n \log^2(k) \log(n))
\end{aligned}$$

■

4.D Threshold based energy-detector

In this section we provide a proof of the Lemma 4.C.1. Let $\vec{u} \in \mathbb{C}^D$ be a complex D-dimensional vector embedded in zero mean white Guassian noise $\vec{w} \in \mathcal{C}N(0, I_{D\times D})$. The energy detector fails to identify the presence of the vector $\vec{u}$, if $||\vec{u} + \vec{w}||^2 < (1+\gamma)D$, where $0 < \gamma < 1$ is a constant. Next, we analyze the probability of this event.

$$
\begin{aligned}
& Pr(||\vec{u}+\vec{w}||^2 < (1+\gamma)D) \\
= \; & Pr(||\sqrt{2}\vec{u}+\sqrt{2}\vec{w}||^2 < 2D(1+\gamma)) \\
\overset{(a)}{=} \; & Pr(\mathcal{N}(\sqrt{2}||\vec{u}||,1)^2 + \chi^2_{2D-1} < 2D(1+\gamma)) \\
< \; & Pr(\chi^2_{2D-1} < 2D(1-\gamma)) + Pr(\mathcal{N}(\sqrt{2}||\vec{u}||,1)^2 < 2D(1+\gamma) - 2D(1-\gamma)) \\
= \; & Pr(\chi^2_{2D-1} < 2D(1-\gamma)) + Pr(\mathcal{N}(\sqrt{2}||\vec{u}||,1)^2 < 4D\gamma), \qquad (4.30)
\end{aligned}
$$

in (a) we did a change of basis such that $\vec{u}/||\vec{u}||$ is one of the basis vectors. Since $\vec{w}$ is circularly symmetric, change of basis does not change its distribution. Using the fact that $\mathcal{N}(0,1)^2 - 1$ is a sub-exponential random variable with parameters $(4,4)$, we obtain a standard tail bound for χ^2_{2D-1} as follows,

$$Pr(\chi^2_{2D-1} < (2D-1)(1-t)) < 2e^{-(2D-1)t^2/8}, \; \forall \; 0 < t < 1$$

Set $t = \frac{2D\gamma-1}{2D-1}$ and using $D > 1.5/\gamma$, we get,

$$Pr(\chi^2_{2D-1} < 2D(1-\gamma)) < 2e^{-D\gamma^2/9}. \qquad (4.31)$$

Now consider the second term in (4.30) as,

$$
\begin{aligned}
& Pr(\mathcal{N}(\sqrt{2}||\vec{u}||,1)^2 < 4D\gamma) \\
< \; & Pr(\mathcal{N}(\sqrt{2}||\vec{u}||,1) < 2\sqrt{D\gamma})) \\
= \; & Pr(\mathcal{N}(0,1) < -(\sqrt{2}||\vec{u}|| - 2\sqrt{D\gamma})) \\
< \; & 2Pr(\mathcal{N}(0,1) < -(\sqrt{2}||\vec{u}|| - 2\sqrt{D\gamma})_+) \\
< \; & \exp\left\{-\left(||\vec{u}|| - \sqrt{2D\gamma}\right)_+^2\right\}. \qquad (4.32)
\end{aligned}
$$

Substituting (4.31) and (4.32) in (4.30), we get,

$$Pr(||\vec{u}+\vec{w}||^2 < (1+\gamma)D) < 2e^{-D\gamma^2/9} + e^{-\left(||\vec{u}|| - \sqrt{2D\gamma}\right)_+^2}$$

■

Chapter 5

Computing a sparse $2D$ discrete-Fourier-transform

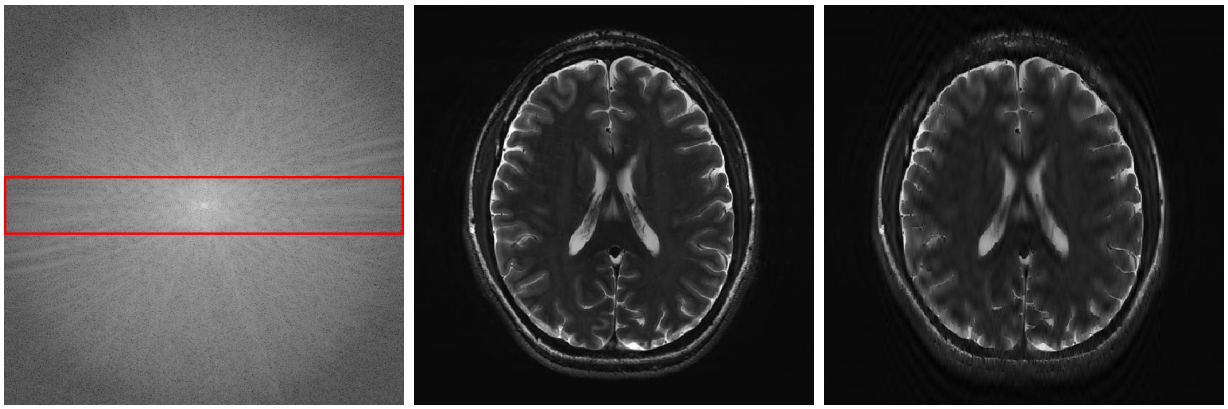
5.1 Introduction

Spectral analysis using the Discrete Fourier Transform (DFT) has been of universal importance in engineering and scientific applications for a long time. The Fast Fourier Transform (FFT) is the fastest known way to compute the DFT. Many applications of interest involve signals, especially $2D$ signals relating to images, e.g. Magnetic-Resonance-Imaging (MRI), video data, surveillance data, satellite imagery, etc., which have a sparse Fourier spectrum. In such cases, a small subset of the spectral components typically contains most or all of the signal energy, with most spectral components being either zero or negligibly small. A naive approach of using a standard $2D$ FFT algorithm to compute the DFT of an $N_0 \times N_1$ dimensional $2D$ signal, requires all the N_0N_1 spatial samples and $O(N_0N_1 \log(N_0N_1))$ arithmetic computations. In applications like MRI, the acquisition time and hence the *sample complexity*, is of vital importance. This motivates us to address the following question in this chapter: In the case when the $2D$-DFT, of an $N_0 \times N_1$ signal is known to have only k non-zero DFT coefficients, where $k << N_0N_1$, how much improvement can one obtain both in terms of the *sample complexity* and the *computational complexity* of computing the DFT?

In Chapter 3, we have addressed the above question for $1D$ signals and proposed a novel FFAST (Fast Fourier Aliasing-based Sparse Transform) framework, which computes an n-length DFT, that has k non-zero coefficients, using only $O(k)$ time-domain samples and $O(k \log k)$ arithmetic operations. In this chapter, we extend the results of Chapter 3 to compute the $2D$-DFT's of $2D$ signals. In particular, we provide an efficient algorithm for computing a k-sparse $2D$-DFT, i.e., has k non-zero coefficients, for the following two classes of $2D$ signals;

- *Signals with co-prime dimensions*: A $2D$ signal of dimension $N_0 \times N_1$, where gcd[1](N_0, N_1)

[1]Greatest common divisor



(a) Log intensity plot of the $2D$-DFT of the original 'Brain' image.

(b) Original 'Brain' image in spatial-domain.

(c) Reconstructed 'Brain' image using the $2D$-FFAST architecture. The total number of Fourier samples used is 60.18%.

Figure 5.1: Application of the $2D$-FFAST algorithm to reconstruct the 'Brain' image acquired on an MR scanner with dimension 504×504. The $2D$-FFAST algorithm reconstructs the 'Brain' image, as shown in Fig. 5.8(c), using overall 60.18% of the Fourier samples of Fig. 5.8(a).

$= 1$. For this class of $2D$ signals, we use the following approach to compute $2D$-DFT. First, we map the problem of computing an $N_0 \times N_1$ $2D$-DFT into that of computing an N, length $1D$-DFT using the Chinese-Remainder-Theorem (CRT), where $N = N_0 N_1$. Then, we use the $1D$-FFAST architecture from Chapter 3 to efficiently compute the $1D$-DFT. Finally, using the Good-Thomas algorithm [36, 74] we recover the desired $2D$-DFT.

- *Signals with equal dimensions*: A $2D$ signal of dimension $N_0 \times N_1$, where $N_0 = N_1$. For this class of $2D$ signals, we propose a new $2D$-FFAST architecture based on the ideas of the $1D$-FFAST framework.

As a motivating example, we demonstrate an application of our proposed $2D$-FFAST algorithm to acquire Magnetic Resonance Imaging (MRI) of the 'Brain' image shown in Fig. 5.8(b). In MRI, recall that the samples are acquired in the Fourier-domain, and the challenge is to speed up the acquisition time by minimizing the number of Fourier samples required to reconstruct the desired spatial-domain image. The FFAST algorithm reconstructs the 'Brain' image acquired on an MR scanner with dimension 504×504, using overall 60.18% of the Fourier samples shown in Fig. 5.8(a). The reconstructed 'Brain' image is shown in Fig. 5.8(c). In Section 5.5.3, we elaborate on the specifics of this simulation result.

The main contribution of this chapter is a $2D$-FFAST algorithm that computes a k-sparse $2D$-DFT of an $N_0 \times N_1$ signal, where the dimensions N_0, N_1 are either co-prime or equal and the non-zero DFT coefficients have uniformly random support, using $O(k)$ samples and in $O(k \log k)$ computations. Our algorithm is applicable for the entire sub-

linear sparsity regime, i.e., $k = O(N^{\delta}), 0 \leq \delta < 1$, where $N = N_0 N_1$. It succeeds with probability approaching 1 asymptotically in k. We emphasize the following caveats. Firstly, our analytical results are probabilistic and are applicable for asymptotic values of N_0, N_1. Secondly, we assume that the $2D$-DFT $\mathbf{X}$ is exactly k-sparse, i.e., has k non-zero DFT coefficients with uniformly random support.

The application of the proposed $2D$-FFAST algorithm to acquire MRI of the 'Brain' image of Fig 5.8(b) leads to some interesting observations. Although the theoretical results of this chapter are applicable to signals that have exactly k non-zero DFT coefficients with uniformly random support, in practice our algorithm is applicable even to realistic signals such as the 'Brain' image of Fig 5.8(b), that are approximately sparse and have a "non-uniform" (or clustered) support for the dominant DFT coefficients.

5.1.1 Related Work

A number of previous works [31, 32, 35, 38, 39] have addressed the problem of computing a 1-D DFT of an n-length signal that has a k-sparse Fourier transform, in sub-linear time and sample complexity. Most of these algorithms achieve a sub-linear time performance by first isolating the non-zero DFT coefficients into different groups/bins, using specific filters or windows that have 'good' (concentrated) support in both, time and frequency. The non-zero DFT coefficients are then recovered iteratively, one at a time. The filters or windows used for the binning operation are typically of length $O(k \log n)$. As a result, the sample and the computational complexity is $O(k \log n)$ or more. The work of [35] provides an excellent tutorial on some of the key ideas used by most sub-linear time sparse FFT algorithms for $1D$ signals in the literature. The FFAST algorithm in Chapter 3 is the first algorithm to compute a k-sparse $1D$-DFT of an n-length signal using $O(k)$ samples and $O(k \log k)$ computations.

While there has been much work in the literature on sub-linear time and sample complexity algorithms for computing a $1D$-DFT, there are very few algorithms designed for $2D$ signals. The algorithm in [34], has $O(k \log^c(N_0 N_1))$ sample and time complexity, for computing a k-sparse $N_0 \times N_1$ $2D$-DFT. In [30], the authors propose sub-linear time algorithms for computing a k-sparse $2D$-DFT for $2D$ signals with equal dimension, i.e., $N_0 = N_1 = \sqrt{N}$, where the dimensions are powers of 2. For an exactly sparse signal, when $k = O(\sqrt{N})$, the algorithm in [30] uses $O(k)$ samples and $O(k \log k)$ time. For a general sub-linear sparsity regime the computational complexity of the algorithm in [30] is $O(k \log k + k(\log \log N)^c)$ for some constant c. The algorithms proposed in [30] succeed with a *constant probability* that does not approach 1. In contrast, the $2D$-FFAST algorithm proposed in this chapter computes a k-sparse $2D$-DFT, for the entire sub-linear regime of sparsity, using $O(k)$ samples and $O(k \log k)$ computations, and succeeds with probability at least $1 - O(1/k)$ that asymptotically (in k) approaches to 1.

In summary, the $2D$-FFAST algorithm is the first to compute an exactly k-sparse $N_0 \times N_1$-point DFT that has all of the following features:

- it has $O(k)$ sample complexity and $O(k \log k)$ computational complexity;
- it covers the *entire* sub-linear regime ($k \propto N^{\delta}, 0 < \delta < 1$), where $N = N_0 N_1$.
- it has a probability of failure that vanishes to zero asymptotically;

The rest of the chapter is organized as follows. In Section 5.2, we provide the description of the problem formulation and the signal model. The main result of the chapter is provided in Section 5.3. In Section 5.4, we describe the $2D$-FFAST framework for both the classes, i.e., co-prime dimensions and equal dimensions, of $2D$ signals using simple examples. We provide extensive simulation results in Section 5.5, for a variety of $2D$ signals, and contrast it with the theoretical claims of this chapter. We also demonstrate an application of the $2D$-FFAST architecture to acquire Magnitude Resonance Imaging (MRI) data of the 'Brain', thus providing an empirical evidence that the $2D$-FFAST architecture is applicable to more realistic signals.

5.2 Problem formulation, notation and preliminaries

Consider a $2D$ signal $\mathbf{x} \in \mathbb{C}^{\mathbf{N_0} \times \mathbf{N_1}}$. A $2D$-DFT of the signal $\mathbf{x}$ is given as:

$$X[\omega_1][\omega_2] = \sum_{t_1, t_2} x[t_1][t_2] e^{\imath 2\pi \omega_1 t_1 / N_0} e^{\imath 2\pi \omega_2 t_2 / N_1}. \tag{5.1}$$

We consider the problem of computing the $2D$-DFT of the signal $\mathbf{x}$. A straightforward computation using a $2D$ fast Fourier transform (FFT) algorithm [4], would require $N = N_0 N_1$ samples and $O(N \log N)$ complex operations. We show that when the transform $\mathbf{X}$ is known to be exactly k-sparse, with a uniformly random support of the non-zero DFT coefficients, one can achieve significant gains in both the number of samples used and the computational complexity of computing the $2D$-DFT. In particular, we propose a $2D$-FFAST framework that computes a k-sparse $2D$-DFT, for the entire sub-linear regime of sparsity, using $O(k)$ samples in $O(k \log k)$ computations and succeeds with probability approaching 1 asymptotically in the number of measurements. The $2D$-FFAST framework proposed in this chapter is an extension of the FFAST architecture from Chapter 3, hence for ease of exposition, we closely follow the notations used in Chapter 3.

We now describe the Chinese-Remainder-Theorem (CRT) which plays an important role in our proposed $2D$-FFAST architecture. For integers a, n, we use $(a)_n$ to denote the modulo operation, i.e., $(a)_n \triangleq a \mod n$.

Theorem 5.2.1 (Chinese-Remainder-Theorem [4])**.** *Suppose $n_0, n_1, \ldots, n_{d-1}$ are pairwise co-prime positive integers and $N = \prod_{i=0}^{d-1} n_i$. Then, every integer 'a' between 0 and $N-1$ is uniquely represented by the sequence $r_0, r_1, \ldots, r_{d-1}$ of its remainders modulo $n_0, \ldots, n_{d-1}$ respectively and vice-versa.*

Notation	Description
N_0, N_1	Ambient signal size in each of the 2 dimensions respectively.
k	Number of non-zero coefficients in the $2D$-DFT $\mathbf{X}$.
δ	Sparsity-index: $k \propto N^{\delta}, 0 < \delta < 1$ and $N = N_0 N_1$.
m	Sample complexity: Number of samples of $\mathbf{x}$ used by the $2D$-FFAST algorithm to compute the $2D$-DFT $\mathbf{X}$.
$r = m/k$	Oversampling ratio: Number of samples per non-zero DFT coefficient.
d	Number of stages in the sub-sampling "front-end" of the FFAST architecture.

Table 5.1: Glossary of important notations and definitions used in the rest of the chapter.

Further, given a sequence of remainders $r_0, r_1, \ldots, r_{d-1}$, where $0 \leq r_i < n_i$, Gauss's algorithm can be used to find an integer 'a', such that,

$$(a)_{n_i} \equiv r_i \quad \text{for } i = 0, 1, \ldots, d-1. \tag{5.2}$$

For example, consider the following pairwise co-prime integers $n_0 = 3, n_1 = 4$ and $n_2 = 5$. Then, given a sequence of remainders $r_0 = 2, r_1 = 2, r_2 = 3$, there exists a unique integer 'a', such that,

$$\begin{aligned} 2 &\equiv a \mod 3 \\ 2 &\equiv a \mod 4 \\ 3 &\equiv a \mod 5 \end{aligned} \tag{5.3}$$

It is easy to verify that $a = 38$ is a unique integer modulo $N = n_0 n_1 n_2 = 60$ that satisfies (5.3).

5.3 Main Result

In this chapter, we propose a novel $2D$-FFAST algorithm, to compute the $2D$-DFT of an $N_0 \times N_1$ signal $\mathbf{x}$, when either $N_0 = N_1$ or $gcd(N_0, N_1) = 1$. The input signal $\mathbf{x}$ is such that the $2D$-DFT, $\mathbf{X}$, has *at most* k complex-valued non-zero coefficients with a uniformly random support in the set $\{(0,0), \ldots, (N_0 - 1, N_1 - 1)\}$. The $2D$-FFAST algorithm computes the k-sparse $N_0 \times N_1$ point $2D$-DFT, with high probability, using $O(k)$ samples of $\mathbf{x}$ in $O(k \log k)$ arithmetic computations. The following theorem states the main result:

Theorem 5.3.1. *For any given $0 < \varepsilon < 1$, there exist (infinitely many) sufficiently large (N_0, N_1), for either $N_0 = N_1$ or $gcd(N_0, N_1) = 1$, such that the $2D$-FFAST algorithm computes the $2D$-DFT $\mathbf{X}$ of a $N_0 \times N_1$ signal $\mathbf{x}$, where $||\mathbf{X}||_0 \leq k$ and $k = \Omega((N_0 N_1)^{\delta})$, with the following properties:*

1. **Sample complexity:** *The algorithm needs $m = r(\delta)k$ samples of the signal $\mathbf{x}$, where the oversampling ratio $r(\delta) > 1$, is a small constant*[2] *that depends on the sparsity index δ;*

2. **Computational complexity:** *The computational complexity of the 2D-FFAST algorithm is $O(k \log k)$, where the constant in big-Oh is small.*

3. **Probability of success:** *The FFAST algorithm successfully computes all the non-zero 2D-DFT coefficients of the signal $\mathbf{x}$, with probability at least $1 - \varepsilon$.*

Proof. The proof for the case when the dimensions N_0, N_1 are co-prime is provided in Section 5.4.1. For the proof of the case when $N_0 = N_1$, please see Appendix 5.A. □

Note, that although Theorem 5.3.1 is for asymptotic values of k, it applies for any signal that has $||\mathbf{X}||_0 \leq k$. Hence, the regime of $\delta = 0$ (esp. constant k) is covered by the 2D-FFAST algorithm designed to operate for any $\delta > 0$, at the expense of being sub-optimal.

5.4 FFAST architecture for $2D$ signals

In this section, we extend the $1D$-FFAST architecture proposed in Chapter 3 to $2D$ signals $\mathbf{x} \in \mathbb{C}^{\mathbf{N_0} \times \mathbf{N_1}}$. We consider two classes of $2D$ signals. First, when the dimensions N_0 and N_1 are co-prime. In this case, there exists a map between a $2D$ signal and a $1D$ signal, such that the $2D$-DFT of a $2D$ signal can be computed using the $1D$-DFT of the mapped $1D$ signal. Hence, one can use the $1D$-FFAST architecture proposed in Chapter 3 *off-the-shelf*, in conjunction with this mapping, to compute the k-sparse $2D$-DFT. The second class of signals that we consider in this chapter is where the dimensions are equal, i.e., $N_0 = N_1$. For this class of signals, we propose a new $2D$-FFAST architecture constructed by extending the $1D$ architecture of Chapter 3. Further, we show that the extended $2D$-FFAST architecture gives almost similar performance guarantees as, for the $1D$ case considered in Chapter 3.

5.4.1 $2D$ signals with co-prime dimensions

Consider an example signal $\mathbf{x} \in \mathbb{C}^{\mathbf{4} \times \mathbf{5}}$, that has a sparse $2D$-DFT. We use the Chinese-Remainder-Theorem (CRT) based forward mapping [36, 74], to obtain a $1D$ signal $\vec{x}$ from the $2D$ signal $\mathbf{x}$ as follows: $1D$ vector $\vec{x}$ is constructed by reading out the samples from $2D$ signal $\mathbf{x}$ in a diagonally downward direction, towards right, starting from top left corner as shown in Fig. 5.2. A uniformly random support in $2D$-DFT corresponds to a uniformly random support in CRT mapped $1D$-DFT. Hence, we use the $1D$-FFAST architecture from Chapter 3, to compute the sparse $1D$-DFT, $\vec{X}$. Then, we perform the reverse mapping of $1D$-DFT $\vec{X}$, to the $2D$-DFT $\mathbf{X}$ using the Good-Thomas algorithm [36, 74]. The elements of

[2] See Fig. 3.2 of Chapter 3, for a relation between the oversampling ratio $r(\delta)$ and the sparsity index δ.

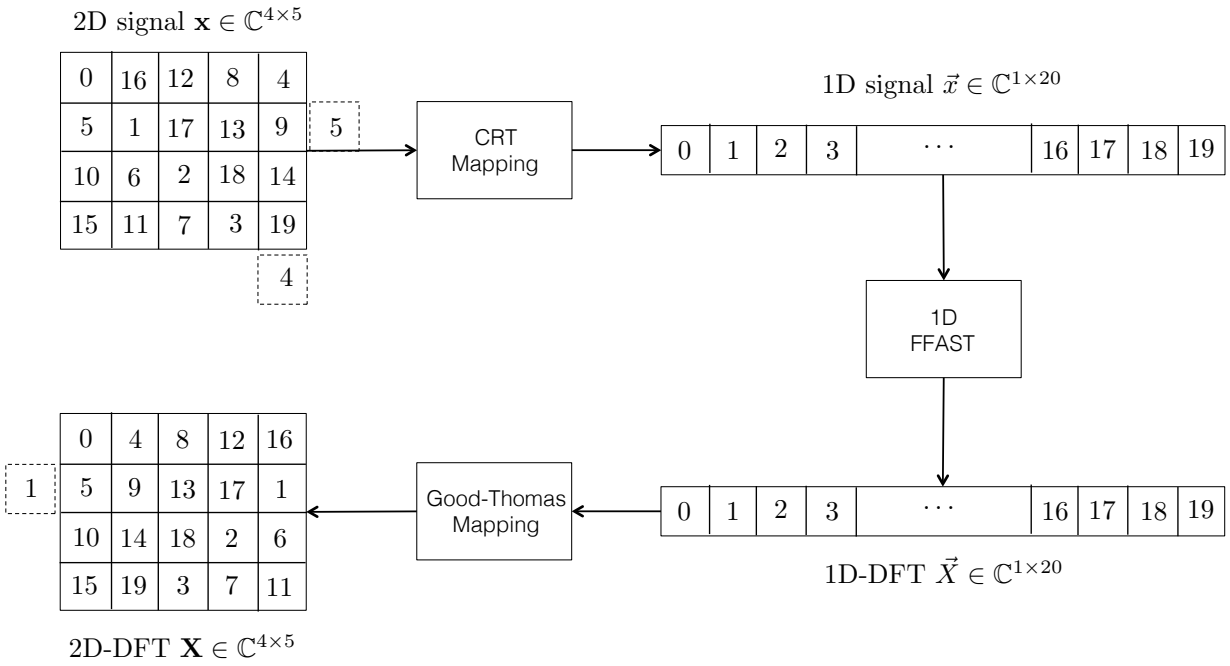


Figure 5.2: Architecture for computing the $2D$-DFT of a $2D$ signal with co-prime dimensions, e.g., $N_0 = 4, N_1 = 5$. First, we use a forward mapping based on the Chinese-Remainder-Theorem (CRT) to convert a $2D$ signal $\mathbf{x}$ in to a $1D$ signal $\vec{x}$. Then, we use our $1D$-FFAST architecture to compute the $1D$-DFT, $\vec{X}$. We transform back the result from $1D$-DFT $\vec{X}$, using a reverse mapping based on Good-Thomas algorithm, to get the $2D$-DFT $\mathbf{X}$.

the $1D$ vector $\vec{X}$ are read out sequentially and placed in a diagonally downward direction, but towards left, starting from the top-left corner as shown in Fig. 5.2.

In [36, 74], the authors show that the forward and the reverse mapping can be done as long as the dimensions are co-prime. Hence, for the signals with co-prime dimensions, the sparse $2D$-DFT can be computed using the $1D$-FFAST algorithm of Chapter 3, along with the forward and reverse mapping from [36, 74].

5.4.2 $2D$ signals with equal dimensions $N_0 = N_1$

In this section we describe a $2D$-FFAST architecture, shown in Fig. 5.3, built on the design principles of the $1D$-FFAST architecture in Chapter 3. Again, we use a simple example to illustrate the $2D$-FFAST framework. Consider a 6×6, $2D$ signal $\mathbf{x}$, such that its $2D$-DFT $\mathbf{X}$ is 4-sparse. Let the 4 non-zero $2D$-DFT coefficients of $\mathbf{x}$ be $X[1][3] = 7, X[2][0] = 3, X[2][3] = 5$ and $X[4][0] = 1$. The $2D$-FFAST 'fonrt-end' sub-samples the input signal and its circularly shifted version through multiple stages d. Each stage, further has $D = 3$ delay paths and is parametrized by a single sampling factor. For example, the FFAST architecture of Fig. 5.3, has $d = 2$ stages and 3 delay (circular shift) paths per stage. Let n_i denote the sampling factor of stage i, e.g., $n_0 = 3$ and $n_1 = 2$, in Fig. 5.3.Stage i of the FFAST front-end samples

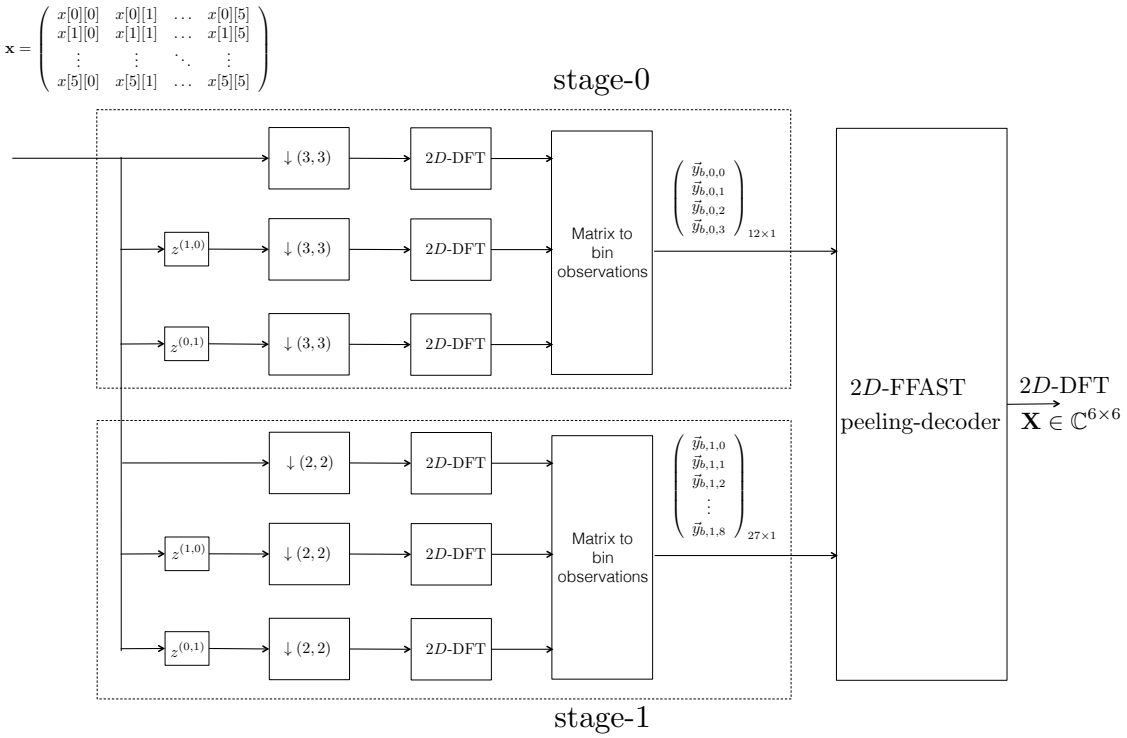


Figure 5.3: A $2D$-FFAST architecture with $d = 2$ stages. In general, the sub-sampling front-end of the $2D$-FFAST architecture consists of 3 or more stages depending on the sparsity index δ, where $k = O(N^{\delta})$ and $N = N_0 N_1$. In this example, we show the $2D$-FFAST architecture with 2-stages only for the purpose of illustration. Each stage further has 3 delay chains and a common sub-sampling factor. The smaller $2D$-DFTs are computed of the output of each sub-sampler. Then, bin-observations are formed by collecting one scalar output from each of the 3 delay chains. The bin-observations are further processed by a 'peeling-decoder' to reconstruct the large $2D$-DFT $\mathbf{X}$.

the input signal $\mathbf{x}$ and its 3 circular shifts, $\mathbf{x^{(0,0)}}, \mathbf{x^{(1,0)}}, \mathbf{x^{(0,1)}}$, by (n_i, n_i), in each of the two dimensions. The FFAST algorithm synthesizes the big $2D$-DFT $\mathbf{X}$, from the short $2D$-DFT of each of the sub-sampled stream, using the peeling-decoder. In Section 5.4.2, we describe in detail, the operation of the block *matrix-to-bin observations*, which essentially groups the output of the short DFTs in a specific way.

Before we describe how to compute the $2D$-DFT of the signal $\mathbf{x}$, using the $2D$-FFAST framework, we review some basic signal processing properties of subsampling-aliasing and circular shifts. In Fig. 5.4, we show a detailed view of stage 0 of the $2D$-FFAST architecture of Fig. 5.3.

- **Sub-sampling and aliasing:** If a $2D$ signal is subsampled in the spatial-domain, its $2D$-DFT coefficients mix together, i.e., alias, in a pattern that depends on the sampling procedure. For example, consider uniform subsampling of $\mathbf{x}$ by 3 in both the dimensions. The sub-sampling operation in the first path or delay chain of Fig. 5.4,

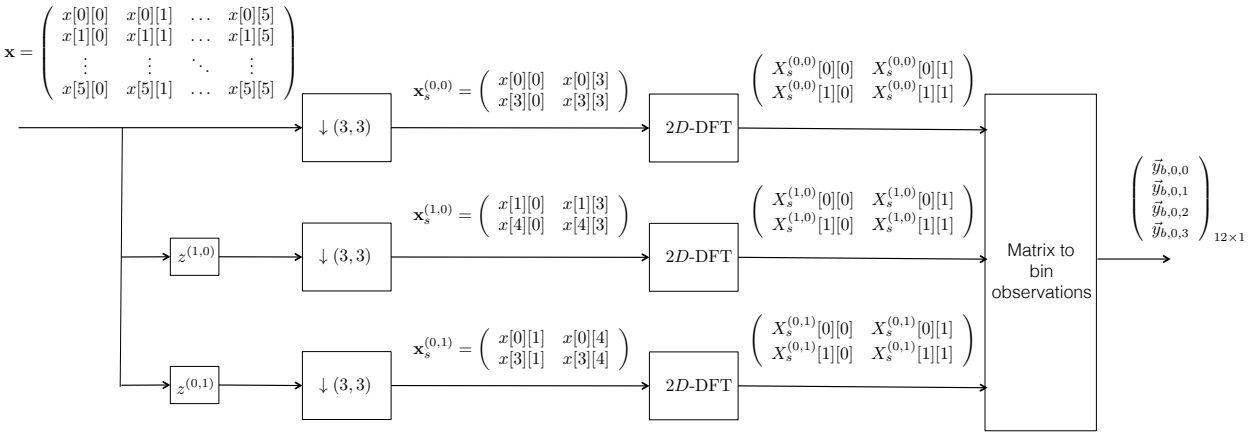


Figure 5.4: A detailed view of stage 0 of the 2D-FFAST architecture of Fig. 5.3. A 2D signal $\mathbf{x}$ and its 2D circularly shifted versions are subsampled by $(3,3)$ to obtain the sub-sampled signals $\mathbf{x_s^{0,0}}, \mathbf{x_s^{1,0}}$ and $\mathbf{x_s^{0,1}}$. Then, the 2D-DFT of the sub-sampled signal is computed and the outputs are collected to form bin-observations.

results in $\mathbf{x_s} = \begin{pmatrix} x[0][0] & x[0][3] \\ x[3][0] & x[3][3] \end{pmatrix}$. Then, the 2$D$-DFT coefficients of $\mathbf{x_s}$ are related to the 2D-DFT coefficients $\mathbf{X}$ as:

$$\begin{aligned}
X_s[0][0] &= X[0][0] + X[2][0] + X[4][0] + X[0][2] + X[2][2] \\
&\quad + X[4][2] + X[0][4] + X[2][4] + X[4][4] \\
&= 4 \\
X_s[0][1] &= X[0][1] + X[2][1] + X[4][1] + X[0][3] + X[2][3] \\
&\quad + X[4][3] + X[0][5] + X[2][5] + X[4][5] \\
&= 5 \\
X_s[1][0] &= X[1][0] + X[3][0] + X[5][0] + X[1][2] + X[3][2] \\
&\quad + X[5][2] + X[1][4] + X[3][4] + X[5][4] \\
&= 0 \\
X_s[1][1] &= X[1][1] + X[3][1] + X[5][1] + X[1][3] + X[3][3] \\
&\quad + X[5][3] + X[1][5] + X[3][5] + X[5][5] \\
&= 7.
\end{aligned}$$

More generally, if the sampling periods of the row and columns of the 2D-signal $\mathbf{x}$ are n_0 and n_1 (we assume that n_0 divides N_0 and n_1 divides N_1) respectively, then,

$$X_s[i][j] = \sum_{i \equiv (a)_{N_0/n_0}, j \equiv (b)_{N_1/n_1}} X[a][b] \tag{5.4}$$

where notation $i \equiv (a)_{N_0/n_0}$, denotes $i \equiv a \bmod (N_0/n_0)$.

- **Circular spatial shift:** A circular shift in the spatial-domain results in a phase shift in the frequency-domain. Consider a circularly shifted signal $\mathbf{x^{(1,0)}}$ (each column is circularly shifted by 1) obtained from $\mathbf{x}$ as $x^{(1,0)}[i][j] = x[(i+1)_{N_0}][j]$. Then the $2D$-DFT coefficients of the shifted signal are given as, $X^{(1,0)}[i][j] = \omega_{N_0}^{i} X[i][j]$, where $\omega_{N_0} = \exp(2\pi\imath/N_0)$. Similarly, the $2D$-DFT coefficients of a circularly shifted sequence $\mathbf{x^{(0,1)}}$ obtained from $\mathbf{x}$ as $x^{(0,1)}[i][j] = x[i][(j+1)_{N_1}]$, are $X^{(0,1)}[i][j] = \omega_{N_1}^{j} X[i][j]$, where $\omega_{N_1} = \exp(2\pi\imath/N_1)$. In general, a circular shift of (s_1, s_2) results in $X^{(s_1,s_2)}[i][j] = \omega_{N_0}^{is_1}\omega_{N_1}^{js_2} X[i][j]$.

Using the above signal processing properties of sub-sampling and spatial circular shift, we compute the relation between the $2D$-DFT coefficients $\mathbf{X}$ and the output of stage 0 (shown in Fig. 5.4) of the $2D$-FFAST front-end. Next, we group the output of the $2D$-FFAST front end into "bin-observation" as follows:

Bin observation

A bin observation is a 3-dimensional vector formed by collecting one scalar output value from each of the 3 delay chains in a stage of the $2D$-FFAST front-end. For example, $\vec{y}_{b,0,0}$ is an observation vector of bin 0 in stage 0 of the $2D$-FFAST front-end and is given by,

$$\vec{y}_{b,0,0} = \begin{pmatrix} X_s^{(0,0)}[0][0] \\ X_s^{(1,0)}[0][0] \\ X_s^{(0,1)}[0][0] \end{pmatrix}. \tag{5.5}$$

Note, that there are total of 4 bins in stage 0 shown in Fig. 5.4. The bins are indexed by a single number as follows: a bin formed by collecting the scalar outputs indexed by (i, j) from each of the delay chains is labelled $(N_0/n_0) \times i + j$, e.g., all the delay chain outputs indexed by $(1, 0)$ form the observation vector of bin 2 of stage 0 (also see Fig. 5.5), and is denoted by

$$\vec{y}_{b,0,2} = \begin{pmatrix} X_s^{(0,0)}[1][0] \\ X_s^{(1,0)}[1][0] \\ X_s^{(0,1)}[1][0] \end{pmatrix}.$$

Next, using the above 6×6 example signal $\mathbf{x}$, we explain how to compute a sparse $2D$-DFT of a signal using decoding over an appropriately designed sparse graph.

Computing a sparse $2D$-DFT via decoding on a sparse-graph

Let the 6×6 example signal $\mathbf{x}$ be processed through the 2-stage $2D$-FFAST architecture of Fig. 5.3. Then, the relation between the resulting bin-observations and the non-zero $2D$-DFT coefficients $\mathbf{X}$ can be computed using the sub-sampling and the circular shift properties

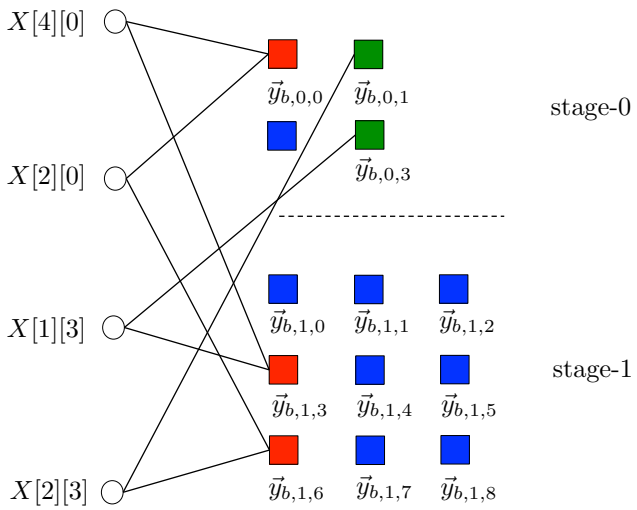


Figure 5.5: A bi-partite graph representing the relation between the bin-observations and the non-zero $2D$-DFT coefficients $\mathbf{X}$. Left nodes in the graph represent the non-zero $2D$-DFT coefficients and the right nodes represent the "bins" (we sometime refer them also as check nodes) with vector observations. An edge connects a left node to a right check node iff the corresponding non-zero $2D$-DFT coefficient contributes to the observation vector at that particular check node, e.g., the $2D$-DFT coefficient $X[4][0]$ contributes to the observation vector of bin 0 of stage 0 and bin 3 of stage 1. A zeroton check node has no left neighbor. A singleton check node has exactly one left neighbor and a multiton check node has more than one left neighbors in the graph.

explained earlier and is graphically represented in Fig. 5.5. Left nodes of the graph in Fig. 5.5 represent the non-zero DFT coefficients and the right nodes represent the "bins" (check nodes) with vector observations. An edge connects a left node to a right check node iff the corresponding non-zero $2D$-DFT coefficient contributes to the observation vector of that particular check node, e.g., after aliasing, due to sub-sampling, the $2D$-DFT coefficient $X[4][0]$ contributes to the observation vector of bin 0 of stage 0 and bin 3 of stage 1.

We define the following:

- **zero-ton**: A bin that has no contribution from any of the non-zero DFT coefficients of the signal, e.g., bin 2 of stage 0 or bin 0 of stage 1 in Fig. 5.5. A zero-ton bin can be trivially identified from its observations.

- **single-ton**: A bin that has contribution from exactly one non-zero DFT coefficient of the signal, e.g., bin 1 of stage 0. Using the signal processing properties the observation vector of bin 1 of stage 0 is given as,

$$\vec{y}_{b,0,1} = \begin{pmatrix} X[2][3] \\ e^{2\pi\imath 2/6} X[2][3] \\ e^{2\pi\imath 3/6} X[2][3] \end{pmatrix}.$$

  The observation of a singleton bin can be used to determine the $2D$ support and the value, of the only non-zero DFT coefficient contributing to that bin, as follows:

– *row support*: The row-support of the non-zero DFT coefficient contributing to a singleton bin can be computed as,

$$2 = \frac{6}{2\pi}\angle y_{b,0,1}[1]y^{\dagger}_{b,0,1}[0] \tag{5.6}$$

– *column support*: The column-support of the non-zero DFT coefficient contributing to a singleton bin can be computed as,

$$3 = \frac{6}{2\pi}\angle y_{b,0,1}[2]y^{\dagger}_{b,0,1}[0] \tag{5.7}$$

– *Value*: The value of the non-zero DFT coefficient is given by the observation $y_{b,0,1}[0]$.

We refer to this procedure as a "ratio-test", hence forth. Thus, a simple ratio-test on the observations of a singleton bin correctly identifies the $2D$ support and the value of the only non-zero DFT coefficient connected to that bin. This property holds for all the singleton bins.

- **multi-ton**: A bin that has contribution from more than one non-zero DFT coefficients of the signal, e.g., bin 0 of stage 0. The observation vector of bin 0 of stage 0 is,

$$\vec{y}_{b,0,0} = \begin{pmatrix} X[4][0] + X[2][0] \\ e^{2\pi\imath 4/6}X[4][0] + e^{2\pi\imath 2/6}X[2][0] \\ X[4][0] + X[2][0] \end{pmatrix} = \begin{pmatrix} 4 \\ -2 + \imath\sqrt{3} \\ 4 \end{pmatrix}.$$

Now if we perform the "ratio-test" on these observations, we get the column support to be 2.3184. Since we know that the column support has to be an integer value between 0 to 5, we conclude that the observations do not correspond to a singleton bin. In Chapter 2, we rigorously show that the ratio-test identifies a multi ton bin almost surely.

Hence, using the "ratio-test" on the bin-observations, the FFAST decoder can determine if a bin is a zero-ton, a single-ton or a multi-ton, almost surely. Also, when the bin is singleton the ratio-test provides the $2D$-support as well as the value of the non-zero DFT coefficient connected to that bin. We use the following peeling-decoder on the graph in Fig. 5.5, to compute the support and the values of the non-zero DFT coefficients of $\mathbf{x}$.

Peeling-Decoder: The peeling-decoder repeats the following steps:

1. Select all the edges in the graph with right degree 1 (edges connected to singleton bins).

2. Remove these edges from the graph as well as the associated left and the right nodes.

3. Remove all other edges that were connected to the left nodes removed in step-2. When a neighboring edge of any right node is removed, its contribution is subtracted from that bins observation.

Decoding is successful if, at the end, all the edges have been removed from the graph. It is easy to verify that performing the peeling procedure on the graph of Fig. 5.5, results in successful decoding with the coefficients being uncovered in the following possible order: $X[2][3], X[1][3], X[4][0], X[2][0]$.

Thus, the $2D$-FFAST architecture of Fig. 5.3, has transformed the problem of computing a $2D$-DFT of a signal $\mathbf{x}$ into that of decoding over a sparse graph of Fig. 5.5, induced by the sub-sampling operations. The success of the peeling-decoder depends on the properties of the induced graph. In Appendix 5.A we show that, if the sub-sampling parameters are chosen carefully, guided by the Chinese-Remainder-Theorem (CRT), the FFAST decoder succeeds with probability approaching 1, asymptotically in k.

5.5 Simulations

In this section we validate the empirical performance of the 2D-FFAST architecture for various settings. First, we provide simulations for synthetic $2D$ signals with an exactly sparse $2D$-DFT and contrast it with the theoretical claims of this chapter. Then we show an application of the $2D$-FFAST architecture to acquire Magnitude Resonance Imaging (MRI) data, which is not exactly sparse and also has a structured, rather than a uniformly random, support for the dominant $2D$-DFT coefficients. Thus, providing an empirical evidence that the $2D$-FFAST architecture is applicable to realistic signals, although the theoretical claims are only for signals with an exactly sparse $2D$-DFT with uniformly random support.

5.5.1 Application of $2D$-FFAST for signals with $N_0 = N_1$, and exactly k-sparse $2D$-DFT

- **Very sparse regime $0 < \delta \leq 1/3$:** We construct a $2D$ signal $\mathbf{x}$ of ambient dimensions $N_0 = N_1 = 2520$. The sparsity parameter k is varied from 70 to 170 which corresponds to the very-sparse regime of $k < O(N^{1/3})$, where $N = N_0 N_1 = 6.35$ million. We use a 4 stage $2D$-FFAST architecture (see Fig. 5.3 for reference). The uniform sampling periods for each of the 4 stages are $(5 \times 7 \times 8)^2$, $(7 \times 8 \times 9)^2$, $(8 \times 9 \times 5)^2$ and $(9 \times 5 \times 7)^2$ respectively. Thus the number of bins/check-nodes in the 4 stages are 9^2, 5^2, 7^2 and 8^2 respectively. Further, each stage has 3 delay sub-streams. Thus, the total number of samples used by the FFAST algorithm for this simulation is $m < 3(9^2 + 5^2 + 7^2 + 8^2) = 657$. We define η as an average number of bins per stage normalized by the sparsity k. For example, when $k = 100$ the value of $\eta = 0.5475$. In Chapter 3, we have shown that for a 4-stage $1D$-FFAST architecture the theoretical threshold $\eta^* = 0.3237$. We

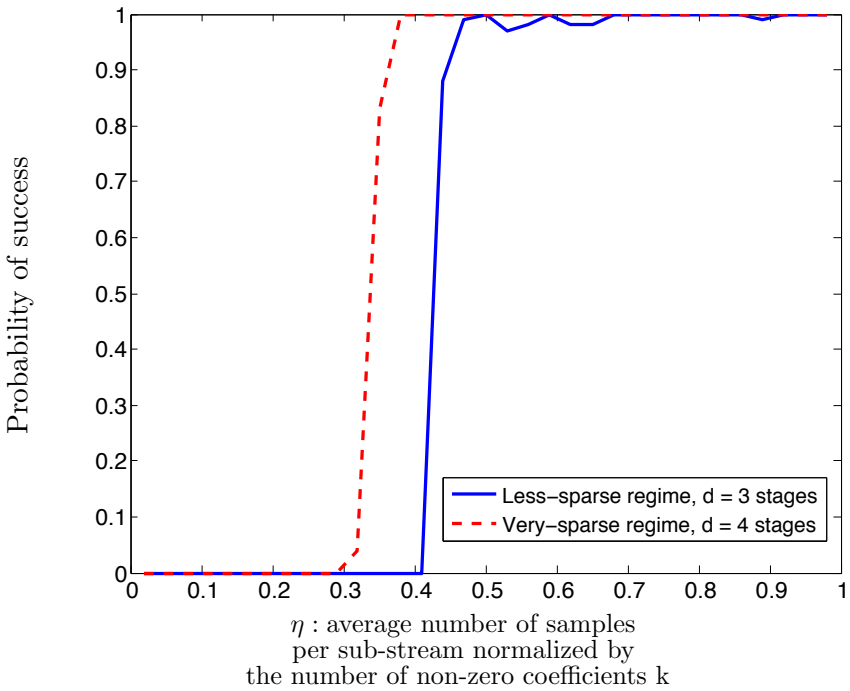


Figure 5.6: The probability of success of the $2D$-FFAST algorithm as a function of η, the average number of bins per stage normalized by the number of non-zero coefficients k. The plot is obtained for the two different sparsity regimes: 1) Very-sparse regime, i.e., $\delta \leq 1/3$. For this regime, a $d = 4$ stage $2D$-FFAST architecture is used; 2) Less-sparse regime, i.e., $1/3 < \delta < 1$. For this regime, a $d = 3$ stage $2D$-FFAST architecture is used. Each point in the plot is obtained by averaging over 100 runs of the simulations. The ambient signal dimension $N_0 \times N_1$, and the number of samples m are fixed in both the simulations, while the number of the non-zero $2D$-DFT coefficients k, is varied to get different values of η. We note that the FFAST algorithm exhibits a threshold behavior with the thresholds $\eta_1 = 0.38$ and $\eta_2 = 0.47$ for the very-sparse and the less-sparse regimes respectively. From Chapter 3, we know that the optimal thresholds for a $d = 4$ and $d = 3$ stage FFAST architecture are $\eta_1^* = 0.3237$ and $\eta_2^* = 0.4073$ respectively. Thus, confirming that the empirical behavior of the $2D$-FFAST architecture is in close agreement with the theoretical results.

observe that the $2D$-FFAST algorithm also shows a threshold behavior and successfully reconstructs the $2D$-DFT for all values of $\eta > 0.38$, which is in close agreement with the theoretical claims.

- **Less sparse regime** $1/3 < \delta < 1$**:** We construct a $2D$ signal $\mathbf{x}$ of ambient dimensions $N_0 = N_1 = 280$. The sparsity parameter k is varied from 2000 to 5000 which corresponds to the less-sparse regime, $O(N^{2/3}) < k < O(N^{3/4})$, where $N = N_0 N_1 = 78400$. We use a $d = 3$ stage $2D$-FFAST architecture (see Fig. 5.3 for reference). The uniform sampling periods for each of the 3 stages are $5, 8$ and 7 respectively. Thus the number of bins/check-nodes in the 3 stages are 35^2, 56^2 and 40^2 respectively. Further, each stage has 3 delay sub-streams. Thus the total number of samples used by the FFAST algorithm for this simulation is $m < 3(35^2 + 56^2 + 40^2) = 17883$. From Chapter 3, for a 3-stage $1D$-FFAST architecture, the theoretical threshold $\eta^* = 0.4073$. We observe

that the $2D$-FFAST algorithm successfully reconstructs the $2D$-DFT for all values of $\eta > 0.47$, which is in close agreement with the theoretical thresholds.

- For each run of the simulation, a $2D$-DFT $\mathbf{X}$ of dimension $N_0 \times N_1$ is generated, with exactly k non-zero coefficients. The support of the non-zero DFT coefficients is chosen uniformly at random from the set $\{0, 1, ..., N_0 - 1\} \times \{0, 1, ..., N_1 - 1\}$. The spatial $2D$ signal $\mathbf{x}$ is then generated from $\mathbf{X}$ using an inverse $2D$-DFT operation. This discrete signal $\mathbf{x}$ is then given as an input to the $2D$-FFAST front-end.
- Each sample point in Fig. 5.6 is generated by averaging over 100 runs of the simulations.
- Decoding is successful if all the $2D$-DFT coefficients are recovered perfectly.

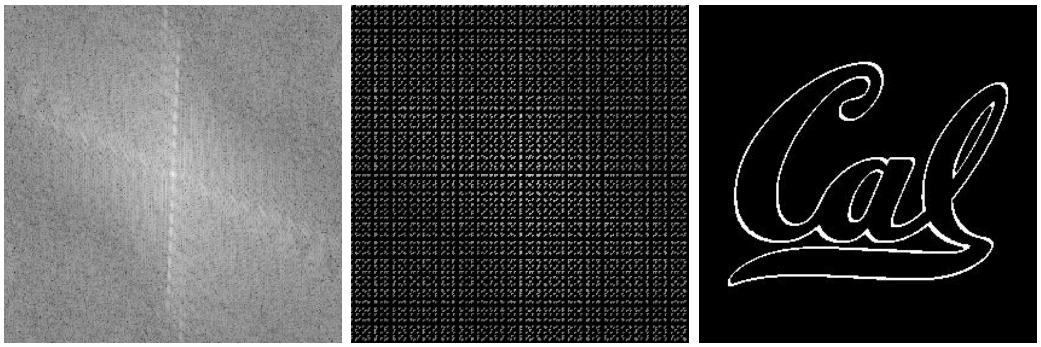
5.5.2 Application of $2D$-FFAST for signals with exactly k-sparse $2D$-DFT but with non-uniform support

The theoretical guarantees of Theorem 5.3.1 are for signals that have an exactly k-sparse $2D$-DFT, with the support of the non-zero DFT coefficients being uniformly random. In this section we empirically show that the proposed $2D$-FFAST architecture works quite well even for signals that have non-uniform support of the $2D$-DFT coefficients. We provide simulation results for two different types of images with the parameters specified as below.

- **'Cal' Image:** The 'Cal' image shown in Fig. 5.7(c), is a synthetic image of size 280×280. The number of the non-zero pixels in the 'Cal' image is $k = 3509$, and the support is non-uniform. Note, since the image is sparse in the spatial-domain, we sub-sample in the frequency-domain, i.e., the input to the FFAST front-end is the $2D$-DFT of the 'Cal' image (see Fig. 5.7(a)). We use a 3 stage $2D$-FFAST architecture with sampling periods $5, 7$ and 8 in each of the 3 stages respectively. The FFAST algorithm perfectly reconstructs the spatial image using $m = 16668$ samples of its Fourier-domain.
- **'House' Image:** The 'House' image shown in Fig. 5.7(f) is created by applying the Canny edge detection algorithm on the commonly used House image in the image processing literature. The 'House' image is cropped to size 247×238. The number of non-zero pixels is $k = 4599$ and the support is non-uniform. Since the dimensions of the image are co-prime, we use the prime factor mapping of Section 5.4.1, and apply the $1D$-FFAST algorithm to perfectly reconstruct the image using $m = 5.46k = 25126$ samples of its Fourier-domain.

5.5.3 Application of the $2D$-FFAST for MR imaging

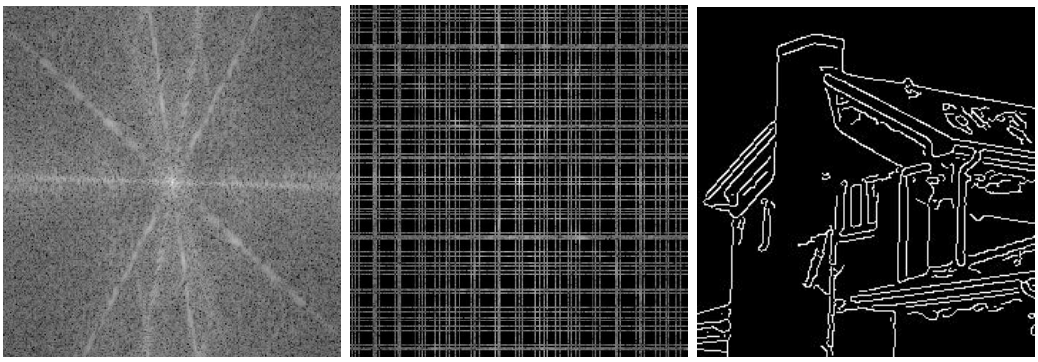
In this section, we apply the $2D$-FFAST algorithm to reconstruct a brain image acquired on an MR scanner with dimension 504×504. In MR imaging the samples are acquired



(a) Log-intensity plot of the $2D$-DFT of the 'Cal' image of dimension 280×280.

(b) $2D$-FFAST subsampled $2D$-DFT of the 'Cal' image with $m = 4.75k = 16668$ samples. The white pixels correspond to the sampled data.

(c) Perfectly reconstructed 'Cal' image using the 2D FFAST algorithm



(d) Log-intensity plot of the $2D$-DFT of the 'House' image of dimension247×238.

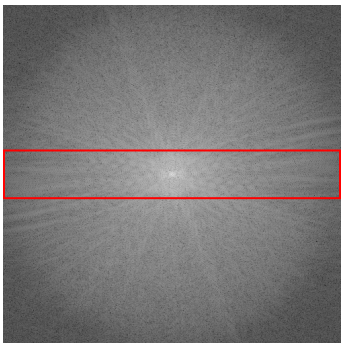
(e) $1D$-FFAST subsampled $2D$-DFT of 'House' image with $m = 5.46k = 25126$ samples.

(f) Perfectly reconstructed 'House' image using prime-factor mapping and the $1D$-FFAST algorithm

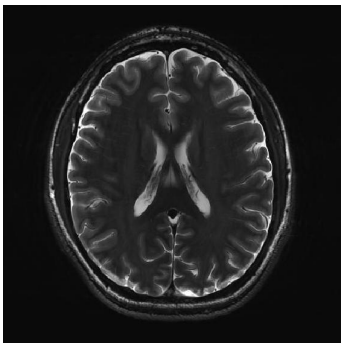
Figure 5.7: The figure shows the performance of the $2D$ and $1D$ FFAST algorithm for signals with exactly-sparse spatial image-domain, with non-uniform support of the non-zero pixels. The signal is sampled in the Fourier-domain and the FFAST algorithm is used to perfectly recover the image-domain data.

in the Fourier-domain and the task is to reconstruct the spatial image from significantly less number of Fourier samples. To reconstruct the full brain image using $2D$-FFAST, we perform the following two-step procedure:

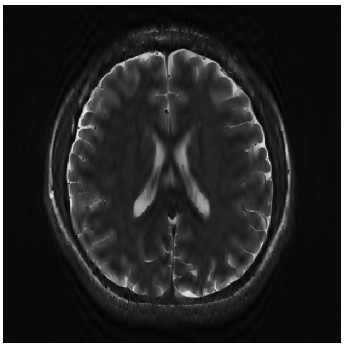
- *Differential space signal acquisition*: We perform a vertical finite difference operation on the image by multiplying the $2D$-DFT signal with $1 - e^{2\pi\imath\omega_0}$. This operation effectively creates an approximately sparse differential image, as shown in Fig. 5.8(e), in spatial-domain and can be reconstructed using FFAST. Note, that the finite difference operation can be performed on the sub-sampled data and at no point we access



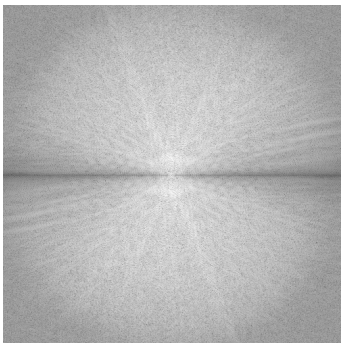
(a) Log intensity plot of the $2D$-DFT of the original 'Brain' image. The red enclosed region is fully sampled and used for the stable inversion.



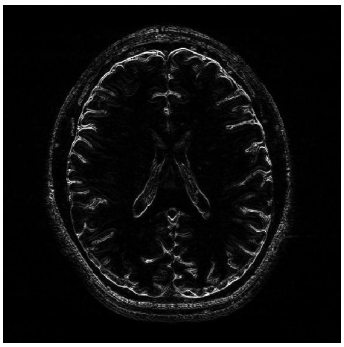
(b) Original 'Brain' image in spatial-domain.



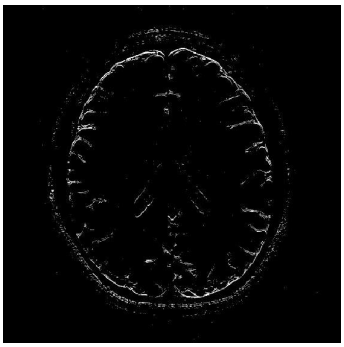
(c) Reconstructed 'Brain' image using the $2D$-FFAST architecture along with the fully sampled center frequencies. The total number of Fourier samples used is 60.18%.



(d) Log intensity plot of $2D$-DFT of the original 'Brain' image, after application of the vertical difference operation.



(e) Differential 'Brain' image obtained using the vertical difference operation on the original 'Brain' image.



(f) Differential 'Brain' image reconstructed using the $2D$ FFAST algorithm from 56.71% of Fourier samples.

Figure 5.8: Application of the $2D$-FFAST algorithm to reconstruct the 'Brain' image acquired on an MR scanner with dimension 504×504. We first reconstruct the differential 'Brain' image shown in Fig. 5.8(e), using $d = 3$ stage $2D$-FFAST architecture with 15 random delays in each of the 3 stages of the FFAST architecture. Additionally we acquire all the Fourier samples from the center frequency as shown, by the red enclosure, in Fig. 5.8(a). Then, we do a stable inversion using the reconstructed differential 'Brain' image of Fig. 5.8(f) and the fully sampled center frequencies of Fig. 5.8(a), to get a reconstructed full 'Brain' image as shown in Fig. 5.8(c). Our proposed two-step acquisition and reconstruction procedure takes overall 60.18% of Fourier samples.

all the input Fourier samples. The differential brain image is then sub-sampled and reconstructed using a 3 stage $2D$-FFAST architecture. Also, since the brain image is approximately sparse, we take 15 delay sub-streams in each of the 3 stages of the $2D$-FFAST architecture, instead of 3 delay sub-streams as in the exactly sparse case. The FFAST algorithm reconstructs the differential brain image using 56.71% of Fourier

samples.

- *Inversion using fully sampled center frequencies*: After reconstructing the differential brain image, as shown in Fig. 5.8(f), we invert the finite difference operation by dividing the $2D$-DFT samples with $1 - e^{2\pi\imath\omega_0}$. Since the inversion is not stable near the center of the Fourier-domain, only the non-center frequencies are inverted. The center region of the $2D$-DFT is fully sampled and used in the inversion process.

- Overall we use a total of 60.18% of Fourier samples to reconstruct the brain image using the $2D$ FFAST algorithm along with the fully sampled center frequencies.

5.A proof of Theorem 5.3.1

In this section we provide a proof of Theorem 5.3.1, for the case when $N_0 = N_1$. In Chapter 3, we have shown that for the $1D$ signals, for all values of the sparsity index $0 < \delta < 1$, and sufficiently large (k, n), where $k = \Omega(n^\delta)$, there exists a $1D$-FFAST architecture, that computes a k-sparse $1D$-DFT $\vec{X}$, using $O(k)$ sample in $O(k \log k)$ computations. The FFAST algorithm succeeds with probability approaching 1, asymptotically in the number of measurements. In order to show a similar result for the $2D$ signals, for the case when $N_0 = N_1$, we use the following approach.

First we show that, for any given sparsity index δ and sufficiently large k, N_0, N_1, with $N_0 = N_1$, there exists a mapping between a $2D$-FFAST architecture and a $1D$-FFAST architecture designed for (δ, k, N). The proof then follows from the results in Chapter 3. For the sake of brevity we show the mapping for $\delta = 2/3$. The mapping extends in a straightforward way for the other values of $0 < \delta < 1$.

Let $\{\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2\}$ be a set of pairwise co-prime integers, such that $\mathcal{P}_i = \mathbf{F} + O(1)$, for an asymptotically large number $\mathbf{F}$. Also, let $N_0 = N_1 = \mathcal{P}_0\mathcal{P}_1\mathcal{P}_2$, and $k = (\mathcal{P}_0\mathcal{P}_1)^2$. Then, $k = O(N^{2/3})$, where $N = N_0N_1$, i.e., $\delta = 2/3$. Consider a 3-stage $2D$-FFAST architecture with the sub-sampling factors $n_0 = \mathcal{P}_2, n_1 = \mathcal{P}_0$ and $n_2 = \mathcal{P}_1$ for the 3 stages respectively. For example, the FFAST architecture in Fig 5.3, has sub-sampling factors $n_0 = 3$ and $n_1 = 2$ for the two stages respectively. Then an input signal $\mathbf{x} \in \mathbb{C}^{\mathbf{N_0} \times \mathbf{N_1}}$, processed through this 3-stage $2D$-FFAST architecture results in the output observations with $(\mathcal{P}_0\mathcal{P}_1)^2, (\mathcal{P}_1\mathcal{P}_2)^2$ and $(\mathcal{P}_2\mathcal{P}_0)^2$ check nodes in the three stages respectively (see Fig. 5.9).

Using the CRT every $2D$ location $(0, 0) \leq (i, j) < (N_0, N_1)$, is uniquely represented by $(r_0^{\text{row}}, r_1^{\text{row}}, r_2^{\text{row}})$ and $(r_0^{\text{col}}, r_1^{\text{col}}, r_2^{\text{col}})$, where $r_\ell^{\text{row}} = i \bmod \mathcal{P}_\ell$ and $r_\ell^{\text{col}} = j \bmod \mathcal{P}_\ell$, for $\ell = 0, 1, 2$. Then, a uniformly random choice of the $2D$-support for a non-zero DFT coefficient corresponds to a uniformly random choice of the remainders $(r_0^{\text{row}}, r_1^{\text{row}}, r_2^{\text{row}})$ and $(r_0^{\text{col}}, r_1^{\text{col}}, r_2^{\text{col}})$. Using the sub-sampling-aliasing and the circular shift properties explained in Section 5.4.2 we obtain a sparse graph, shown in Fig. 5.9(a), representing the relation between the k non-zero DFT coefficients of the $2D$ signal $\mathbf{x}$, and the bin-observations. Note

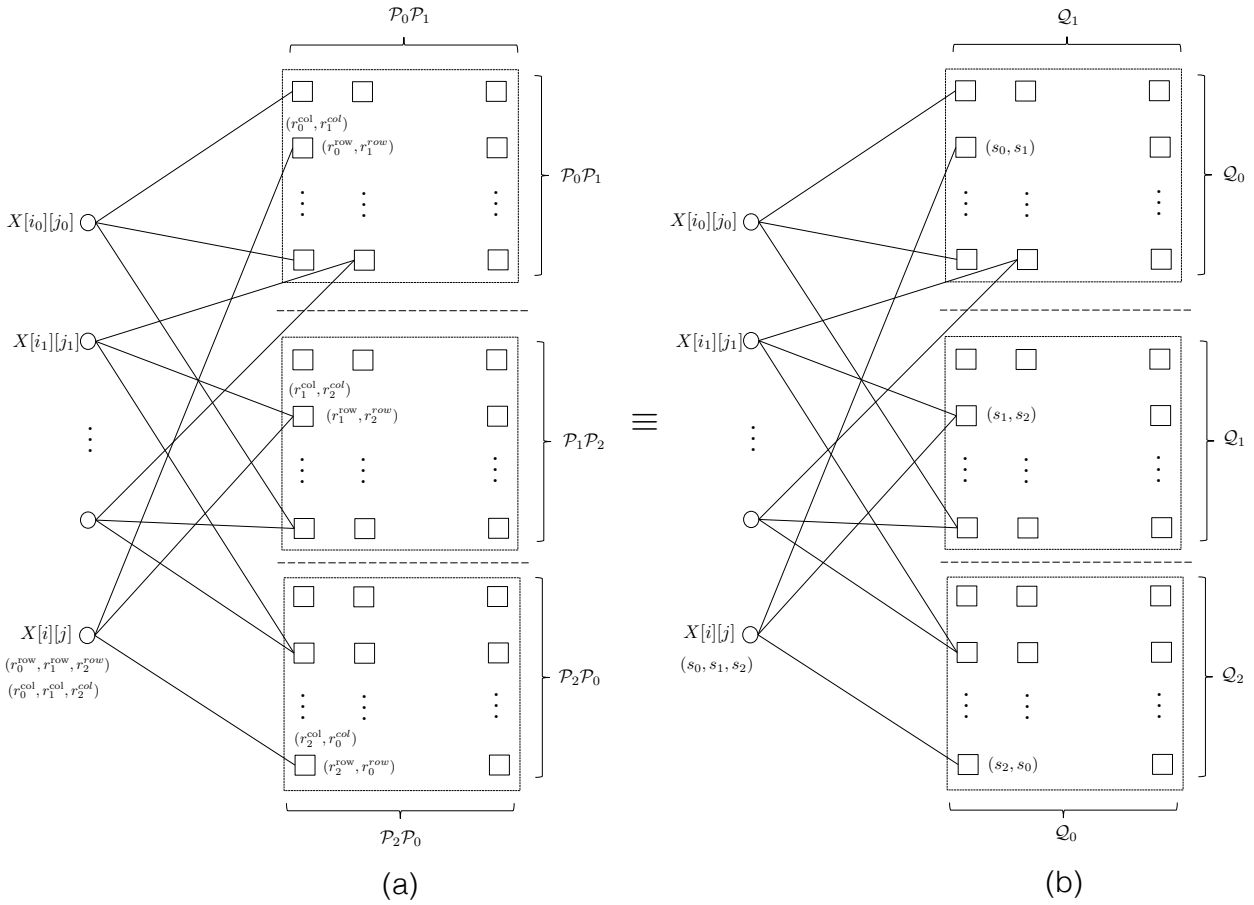


Figure 5.9: A bi-partite graph representation of relation between the non-zero $2D$-DFT coefficients, of a $2D$ signal $\mathbf{x}$, and the observations of a 3-stage $2D$-FFAST architecture. (a) A non-zero DFT coefficient with support (i, j) is indexed by a 6-tuple $((r_0^{\text{row}}, r_1^{\text{row}}, r_2^{\text{row}}), (r_0^{\text{col}}, r_1^{\text{col}}, r_2^{\text{col}}))$, where $r_\ell^{\text{row}} = i \bmod \mathcal{P}_\ell$ and $r_\ell^{\text{col}} = j \bmod \mathcal{P}_\ell$, for $\ell = 0, 1, 2$. Each check node is indexed by a quadruplet, e.g., check node in stage 0 is indexed by $((r_0^{\text{row}}, r_1^{\text{row}}), (r_0^{\text{col}}, r_1^{\text{col}}))$. A non-zero DFT coefficient with an index $((r_0^{\text{row}}, r_1^{\text{row}}, r_2^{\text{row}}), (r_0^{\text{col}}, r_1^{\text{col}}, r_2^{\text{col}}))$ is connected to a check node $((r_0^{\text{row}}, r_1^{\text{row}}), (r_0^{\text{col}}, r_1^{\text{col}}))$ in stage 0. (b) A non-zero DFT coefficient with support (i, j) is indexed by a triplet (s_0, s_1, s_2), where $s_\ell = r_\ell^{\text{row}}\mathcal{P}_\ell + r_\ell^{\text{col}}$, for $\ell = 0, 1, 2$. Each check node is indexed by a doublet, e.g., check node in stage 0 is indexed by (s_0, s_1). A non-zero DFT coefficient with an index (s_0, s_1, s_2) is connected to a check node (s_0, s_1) in stage 0.

that a non-zero DFT coefficient with a $2D$ support (i, j) is connected to a check node indexed by $(r_0^{\text{row}}, r_1^{\text{row}})$ and $(r_0^{\text{col}}, r_1^{\text{col}})$ in stage 0.

Let, $\mathcal{Q}_\ell = \mathcal{P}_\ell^2$ for $\ell = 0, 1, 2$. Then, any doublet $(0, 0) \leq (i, j) < (\mathcal{P}_\ell, \mathcal{P}_\ell)$ can be uniquely represented by an integer $q = i\mathcal{P}_\ell + j$. Using this mapping, we convert every doublet $(r_\ell^{\text{row}}, r_\ell^{\text{col}})$ into a single number $s_\ell = r_\ell^{\text{row}}\mathcal{P}_\ell + r_\ell^{\text{col}}$, for $\ell = 0, 1, 2$. Thus, each check node can now be represented by a doublet instead of a quadruplet and each non-zero DFT coefficient can now be indexed by a triplet instead of a 6-tuple. For example, a check node, in stage 0, with a quadruplet index $((r_0^{\text{row}}, r_1^{\text{row}}), (r_0^{\text{col}}, r_1^{\text{col}}))$ can be represented by a doublet (s_0, s_1). The re-arranged bi-partite graph with this new labeling is shown in Fig 5.9(b).

The bi-partite graph of Fig 5.9(b) has k left nodes and $\mathcal{Q}_0\mathcal{Q}_1+\mathcal{Q}_1\mathcal{Q}_2+\mathcal{Q}_2\mathcal{Q}_0$ right nodes. Further, each left node has one uniformly random neighbor in each of the 3 stages. This graph is a member of the ensemble of bi-partite graphs generated by a 3-stage $1D$-FFAST architecture, in Chapter 3, designed for the following parameters: $\{\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2\}$ is a set of co-prime integers such that $\mathcal{Q}_\ell = O(\mathbf{F}^2)$, $n = \prod_{\ell=0}^{2} \mathcal{Q}_\ell$ and $k = \mathcal{Q}_0\mathcal{Q}_1$, i.e., $\delta = 2/3$. Thus, the proof follows from the results of Chapter 3, for $\delta = 2/3$. ■

Chapter 6

Sparse multivariate polynomial regression

6.1 Introduction

Polynomial interpolation is a well-studied and important problem that frequently occurs in fields like computer algebra and symbolic computation. It is an important step in computational operations such as calculating the GCD (greatest common divisor) of two numbers which arises in many applications. Further, some of the machine learning applications [40, 48] use polynomial regression methods to estimate prediction functions. In general, a polynomial interpolation is the interpolation of a given data set by a polynomial. In most cases the data set consists of some given points and the task is to find a polynomial which fits these these points. In other cases, we may have the flexibility of obtaining evaluations of an underlying unknown polynomial at specific data points of interest and discover the unknown polynomial using these evaluations. In this chapter, we are interested in a special case of the polynomial interpolation problem when the polynomial is known to be of *high-degree* but is *sparse*, i.e., the number of the non-zero coefficients of the polynomial is very small compared to the degree of the polynomial.

In many modern day applications, it is common for the features/variables to be regressed to run into tens of thousands of variables. Executing a brute force polynomial regression algorithm for such dimensions is practically impossible. On other hand it is often the case that the feature space has some additional structure and the regressed polynomial is *sparse*. Motivated by this observation, we are interested in developing efficient interpolation algorithms that exploit the sparse structure of the polynomial to be regressed to reduce the *number of evaluation data points* as well as the *computations* required to determine the polynomial.

In this chapter, we focus on the case where the field is either complex or a finite field. The key idea is to obtain evaluations of the underlying polynomial over the roots of unity, for example in the case of the complex field on the unit circle, and transform the problem

of sparse polynomial interpolation to that of computing a sparse large (degree of the polynomial) dimension discrete Fourier transform (DFT) of the evaluation points. Then use the algorithm proposed in Chapter 3 for computing a sparse DFT to solve the interpolation problem using these evaluations.

Our results can be summarized as follows. We show that, under certain conditions, a polynomial with highest degree n having at most k non-zero coefficients, where $k = o(n)$, can be interpolated using $4k$ or less[1] number of carefully chosen evaluations in $O(k \log k)$ operations for the complex field or $O(k \log k \log^2 q)$ operations in the case of a finite field $\mathbb{F}_q$. This can be contrasted with the best known algorithms in the literature (see Table 6.1). Note that for interpolating a sparse polynomial over the complex field all the known algorithms in the literature require the number of computational operations to be a function of the maximum degree n. To the best of our knowledge the algorithm proposed in this chapter is the first to have *both the evaluation complexity and the computational complexity a function of k and independent of the maximum degree n*. This can be of significant advantage especially when $k << n$. As a concrete example, consider a polynomial of degree $n = 10^6$, over the complex field, with some $k = 200$ non-zero coefficients. Then our proposed sparse polynomial interpolation algorithm computes all the non-zero coefficients from 600 carefully chosen evaluations and $\approx 600 \log(100)$ computations (where we assumed that a k-length DFT can be computed in $k \log k$ complex operations). We also extend our results to the sparse multivariate polynomial interpolation problem using a method known as Kronecker substitution that maps a multivariate polynomial to a univariate polynomial.

We emphasize the following caveats. Our analytical results are probabilistic, and are applicable for asymptotic regime of k, with probability of success approaching 1 asymptotically. The probability space is of uniformly random support of the non-zero coefficients of the polynomial. This can be contrasted with the Ben Or and Tiwari algorithm [2] that deterministically works for any support of the non-zero coefficients and for all values of k, n, but requires $O(k^2 \log^2 k + k^2 \log n)$ operations.

The rest of the chapter is organized as follows. In Section 6.2, we set up the notations and discuss the problem setup. Section 6.3 discusses the main contributions of our work. In Section 6.4, we provide a brief overview of the related work and contrast it with our results. Section 6.5 discusses the mathematical preliminaries and provides a brief review of an algorithm proposed in Chapter 3, for computing a sparse DFT. In Sections 6.6 and 6.7, we discuss our results for the complex and finite fields.

[1]The number of evaluations used by our algorithm depend on the sparsity index $0 < \delta < 1$, that relates the number of the non-zero coefficients k and the degree n of the polynomial, as $k = O(n^\delta)$. For the entire range of practical interest of sub-linear sparsity, i.e., $0 < \delta < 0.99$, the evaluation complexity of our algorithm is $4k$. For the other values of sparsity index δ, the evaluation complexity is $O(k)$.

6.2 Problem Setup and notations

Let $f(x_1, \cdots, x_t) \in \mathbb{F}[x_1, \cdots, x_t]$ be a multivariate polynomial in t variables defined over a field $\mathbb{F}$, with the highest degree of each variate less than N, i.e.,

$$f(x_1, \cdots, x_t) = \sum_{(i_1, \cdots, i_t) \in [0, \cdots, N-1]^t} c_{i_1 \cdots i_t} x_1^{i_1} x_2^{i_2} \cdots x_t^{i_t}. \tag{6.1}$$

We assume that at most k of the coefficients $\{c_{i_1 \cdots i_t}\}$ in (6.1) are non-zero, and the support of the non-zero coefficients is uniformly random, i.e., f is a k-sparse polynomial with uniformly random support. Then the *sparse multivariate polynomial interpolation* problem is one of determining all the coefficients of the polynomial f, from some set of evaluation points of the polynomial. In this chapter, we assume that one can specify the input values at which the underlying polynomial is evaluated. A special case of a multivariate polynomial interpolation problem is the univariate case, i.e., $t = 1$. Let $f(x) \in \mathbb{F}[x]$ be a k-sparse univariate polynomial in x, defined over the field $\mathbb{F}$, with degree less than n, i.e.,

$$f(x) = \sum_{i=0}^{n-1} c_i x^i, \tag{6.2}$$

where at most k of the coefficients $\{c_i\} \in \mathbb{F}$ are non-zero and the task is to determine all the coefficients of the polynomial f, from some set of evaluation points of the polynomial. In Section 6.5.1, we show that a sparse multivariate interpolation problem can be mapped into a sparse univariate interpolation problem. We use this mapping to extend a sparse univariate interpolation algorithm to the multivariate case under certain conditions. Therefore, in the rest of the chapter we focus mostly on the sparse univariate polynomial interpolation problem.

The performance of a sparse univariate polynomial interpolation algorithm can be evaluated using various metrics. We list below some of the performance metrics that can be used to compare different algorithms.

- **Sample complexity** $[m]$**:** The number of polynomial evaluations required by an algorithm to determine all the non-zero coefficients.
- **Computational complexity:** The number of algebraic operations required by an algorithm to determine the non-zero coefficients. We assume the algebraic RAM model where the operations $+, -, \times, \div$ are considered as unit step operations.
- **Reliability:** If the algorithm is probabilistic or assumes a probabilistic model for the input sparse polynomials, then reliability is the probability of determining all the non-zero coefficients correctly over the appropriate probability space.
- **Robustness:** Reliability performance of the algorithm in the presence of either additive noise in the evaluations or finite machine precision arithmetic.

6.3 Main Contributions

We summarize below the main contributions of this chapter. We propose an efficient algorithm to determine the coefficients of a sparse univariate polynomial, over the complex or a finite field, given evaluations of the polynomial at known locations. The algorithm can be extended to interpolate a sparse multivariate polynomial using the mapping described in Section 6.5.1. The algorithm has the following features on the performance metrics of interest.

- **Sample complexity:** It is well known that $2k$ evaluations are sufficient to uniquely determine the coefficients of a k-sparse univariate polynomial [2]. Our algorithm requires $O(k)$ evaluations, where the constant in big-oh is small. For example, the number of the evaluations required is less than $4k$, for any $k < O(n^{0.99})$. This is slightly more than in [2] but significantly lesser than $O(nk)$ as required by Zippel [80].

- **Computational complexity:** The computational complexity of our proposed algorithm is dependent on the field over which the polynomials are defined. For a sparse polynomial interpolation problem over the complex field, the computational complexity of our proposed algorithm is $O(k \log k)$, in contrast the computational complexity of Ben Or and Tiwari's algorithm is $O(k^2 \log^2 k + k^2 \log n)$ and Zippel's algorithm is $O(nk^3)$. To the best of our knowledge, this is the first result we know of in the literature that has computational complexity independent of n, the maximum degree of the polynomial. This can be a significant advantage when $k << n$, which is the case in many applications. For a sparse polynomial interpolation problem over a finite field, the computational complexity of our proposed algorithm is $O(k \log k \log^2 q)$, which is so far the best in the literature (see Table 6.1).

- **Reliability:** Our results are probabilistic in nature, like that of Zippel [80], wherein all the non-zero coefficients of the polynomial are correctly determined with high probability. The proposed algorithm reliably recovers all the coefficients correctly with probability approaching 1 asymptotically in k. In contrast, algorithms like that of Ben Or and Tiwari always determines the coefficients of the sparse polynomial deterministically, but are computationally inefficient. Thus, our algorithm trades-off the computational complexity at the expense of probabilistic guarantees.

- **Parallelizable:** Our proposed algorithm interpolates the non-zero coefficients of the sparse polynomial using an iterative peeling-style recovery algorithm. The peeling-style iterative algorithm, as described in Section 6.5.3, is immensely parallelizable and can result in significant speed up on a multi-core machine.

- **Robustness to noise and finite precision arithmetic:** The algorithm of Ben Or and Tiwari [2], is based on *finding roots of polynomials* which is known to be very *ill-conditioned* numerically. In contrast, an extended version of our algorithm is robust to

Algorithm	Deterministic (D) Random (R)	Sample Complexity	Computational Complexity	Field
Ben Or & Tiwari	D	$2k$	$O(k^2(\log^2 k + \log n))$	$\mathbb{C}$
Zippel	R	$O(nk)$	$O(nk^3)$	$\mathbb{C}$
Ours	R	$O(k)$	$O(k \log k)$	$\mathbb{C}$
Alg.	Deterministic (D) Random (R)	Sample Complexity	Computational Complexity	Field
Garg & Schost I	D	$O(k^2 \log n)$	$O(k^4 \log^2 n)$	$\mathbb{F}_q$
Garg & Schost II	R	$O(k \log n)$	$O(k^3 \log^2 n)$	$\mathbb{F}_q$
Roche	R	$O(\log n)$	$O(k^2 \log^2 n)$	$\mathbb{F}_q$
Javadi & Monagan	R	$O(k)$	$O(nk \log k)$	$\mathbb{F}_q$
Ours	R	$O(k)$	$O(k \log k \log^2 q)$	$\mathbb{F}_q$

Table 6.1: Comparison of algorithms for sparse polynomial interpolation.

additive noise or finite precision arithmetic, albeit at the cost of increased evaluation and the computational complexity.

6.4 Related work

There are many algorithms in the literature for interpolating polynomials starting with basic Lagrangian interpolation [77]. However, not all of these algorithms exploit the sparsity structure of the polynomial for computational efficiency, which is important when dealing with high-degree polynomials. The algorithms in the literature can be categorized as either deterministic or random and are applicable for either the complex field or a finite field or both. A summary of all these algorithms is provided in Table 6.1.

The earliest work is that of Zippel [80], who proposed a probabilistic algorithm to interpolate a k-sparse polynomial defined over the complex field with maximum degree n, using $O(nk)$ evaluations and $O(nk^3)$ computations. Ben Or and Tiwari [2] proposed a deterministic interpolation algorithm that requires only $2k$ evaluations but has a computational complexity of $O(k^2(\log^2 k + \log n))$.

The works of Kaltofen and Yagati [46, 47], Garg and Schost [28], Roche [68], Javadi and Monagan [45] consider the problem of designing an efficient sparse polynomial interpolation algorithm for polynomials over finite fields. The main challenge of operating in finite fields is the computation of the discrete-logarithm which is efficient only under certain constraints [63]. There are also works that consider finite bit precision effects [57, 68].

In this work, we focus mainly on the setup similar to that of Ben Or and Tiwari [2], wherein the assumption is that we can get exact evaluations of the polynomials at desired values of the variable. We first propose a sparse univariate polynomial interpolation al-

gorithm, for polynomials over the complex field, that has *evaluation complexity* $O(k)$ *and computational complexity* $O(k \log k)$*, i.e., independent of the maximum degree* n. This contrasts all existing algorithms in the literature that have computational complexities which are a function of n. We then extend our results to interpolate sparse univariate polynomials over finite fields, using efficient algorithms for computing discrete logarithms proposed in [63], for a certain class of finite fields.

6.5 Preliminaries

In this section, we first provide a mapping of a multivariate polynomial interpolation problem to a univariate polynomial interpolation, which we further map to the problem of computing a sparse DFT. Then, we briefly review an efficient algorithm proposed in Chapter 3 for computing a k-sparse n-length DFT.

6.5.1 Mapping multivariate polynomial interpolation to univariate polynomial interpolation.

Every multivariate polynomial can be mapped to a unique univariate polynomial. In particular, for a given multivariate polynomial, we can find a univariate polynomial such that each monomial term in the multivariate polynomial can be mapped to a unique term in the univariate polynomial. This mapping can be achieved using the method of *Kronecker Substitution*. The following theorem illustrates the mapping from a multivariate polynomial to a univariate polynomial.

Theorem 6.5.1. *(Kronecker, 1882:) Let* $\mathbb{F}$ *be a field and* $t, N \in \mathbb{N}$*. For any polynomial* $f \in \mathbb{F}[x]$*, with degree less than* N^t*, there exists a unique polynomial* $\widehat{f} \in \mathbb{F}[x_1, \cdots, x_t]$ *with partial degrees less than* N*, such that,*

$$f(x) \quad = \quad \widehat{f}(x, x^N, x^{N^2}, \cdots, x^{N^{t-1}}). \tag{6.3}$$

For many problems with multivariate polynomials, making such a substitution allows the use of univariate algorithms. This holds true in the case of polynomial interpolation algorithms as well.

The Kronecker substitution corresponds to writing the exponent of each monomial term of the univariate polynomial in radix-N representation, which gives the corresponding term in the multivariate polynomial. This makes use of the fact that every non-negative integer less than N^t has a unique t-tuple representation in the radix-N representation. For example consider the following bivariate polynomial,

$$\widehat{f}(x_1, x_2) \quad = \quad 6x_1^{11}x_2 + 5x_1x_2^3 - 3. \tag{6.4}$$

Here $N = 12$. The corresponding univariate polynomial is given by,

$$f(x) = \widehat{f}(x, x^{12}) \quad = \quad 6x^{23} + 5x^{37} - 3. \tag{6.5}$$

The mapping from $f(x)$ to $\widehat{f}(x_1, x_2)$ is obtained by writing each exponent of the univariate polynomial $f(x)$ in the radix representation with basis $N = 12$. The radix representation of the exponents in base 12 is given below,

$$23 \quad = \quad 11(12^0) + 1(12^1), \tag{6.6}$$

$$37 \quad = \quad 1(12^0) + 3(12^1). \tag{6.7}$$

Thus, one can see that the exponents of the monomial terms in a multivariate polynomial are the coefficients of the radix representation in basis N. Hence, one could adapt any univariate polynomial interpolation algorithm to a multivariate interpolation algorithm by evaluating the coefficients of the Kronecker substituted univariate polynomial and map the coefficients and exponents back to the multivariate polynomial. Note that the dimension of the maximum degree of the univariate polynomial is increased exponentially in the number of variables. However, since the sample complexity of our algorithms depend only on the number of non-zero coefficients k, which remains unchanged after the mapping, the increase in the degree does not affect the sample complexity of our algorithms. As for the computational complexity, for the case when the polynomials are over the complex field it remains unchanged, while for the polynomials over finite fields the increase is quadratic in logarithm of the field size.

6.5.2 Univariate polynomial interpolation is equivalent to computing a discrete-Fourier-transform

In this section, we discuss how the polynomial interpolation problem can be mapped into that of computing the DFT of an appropriately defined time-domain signal. The mapping is obtained by evaluating the underlying unknown polynomial at different powers of the complex root of unity. Consider a k-sparse univariate polynomial with maximum degree n, over some field $\mathbb{F}$. We assume that the field is such that, there exists an element $\alpha \in \mathbb{F}$ that has order n, i.e. n is the smallest positive integer for which $\alpha^n = 1$. For example, in the case of the complex field, $\alpha = e^{\imath 2\pi/n}$. Note that no such element exists in the real field for $n > 2$. Then, evaluating the underlying unknown polynomial at different powers of α, we obtain the following:

$$f(\alpha^j) \quad = \quad \sum_{i=0}^{n-1} c_i \alpha^{ij}. \tag{6.8}$$

Writing this in matrix form, we get:

$$\begin{bmatrix} f(\alpha^0) \\ f(\alpha^1) \\ \vdots \\ f(\alpha^{(n-1)}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \alpha & \dots & \alpha^{n-1} \\ & \vdots & & \\ 1 & \alpha^{(n-1)} & \dots & \alpha^{(n-1)^2} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}. \tag{6.9}$$

In the above equation, one can view the vector $(c_0, \dots, c_{n-1})$ as the DFT of a time-domain signal $(f(\alpha^0), f(\alpha^1), \dots, f(\alpha^{n-1}))$. Thus, the univariate polynomial interpolation problem is equivalent to computing the DFT of the given time-domain signal, i.e., evaluations at $\{\alpha^i\}_{i=0}^{n-1}$, in an efficient manner. When there is no additional structure imposed on the DFT coefficients, the Fast Fourier Transform (FFT) is the best known algorithm that computes the DFT coefficients using n evaluations and $O(n \log n)$ computations. However, given the additional sparsity constraint that only k out of n DFT coefficients are non-zero, one can potentially design algorithms that are more efficient than the FFT. There have been many recent algorithms targeted towards computing the sparse DFT from a subset of the time-domain signal. In this chapter, we provide a sparse univariate polynomial interpolation algorithm that is based on the algorithm in Chapter 3 for computing a k-sparse DFT using $O(k)$ samples and $O(k \log k)$ arithmetic computations. The algorithm in Chapter 3 is referred to as FFAST and stands for Fast Fourier Aliasing-based Sparse Transform.

Next, for sake of completeness we provide a brief overview of the FFAST architecture and the algorithm. Further, for ease of exposition we use the complex field to explain the ideas underlying the FFAST architecture, which further generalize to finite fields.

6.5.3 Computing a sparse DFT using the FFAST algorithm

In this section, we describe the FFAST sub-sampling "front-end" architecture, as shown in Fig. 6.1, as well as the associated "back-end" FFAST peeling-decoder to compute a k-sparse n-length DFT over the complex field. We use a simple example to illustrate the FFAST sub-sampling front-end and the backend. Consider a 20-point discrete-time signal $\vec{x} = (x[0], \dots, x[19])$, such that its 20-point DFT $\vec{X}$, is 5-sparse. Let the 5 non-zero DFT coefficients of the signal $\vec{x}$ be $X[1] = 1, X[3] = 4, X[5] = 2, X[10] = 3$ and $X[13] = 7$. The FFAST sub-sampling front-end shown in Fig. 6.1, samples the input signal and its circularly shifted version to get 2 sub-sampling paths in each stage. The output of the FFAST sub-sampling front-end is then obtained by computing short DFT's of each of the sub-sampled data as shown in Fig. 6.1. Next, we group the output of the FFAST subsampling front-end into "bin-observations" as follows:

Bin observation

A bin-observation is a 2-dimensional vector formed by collecting one scalar output value from each of the 2 sub-sampling paths in a stage. For example, $\vec{y}_{b,0,1}$ is an observation vector

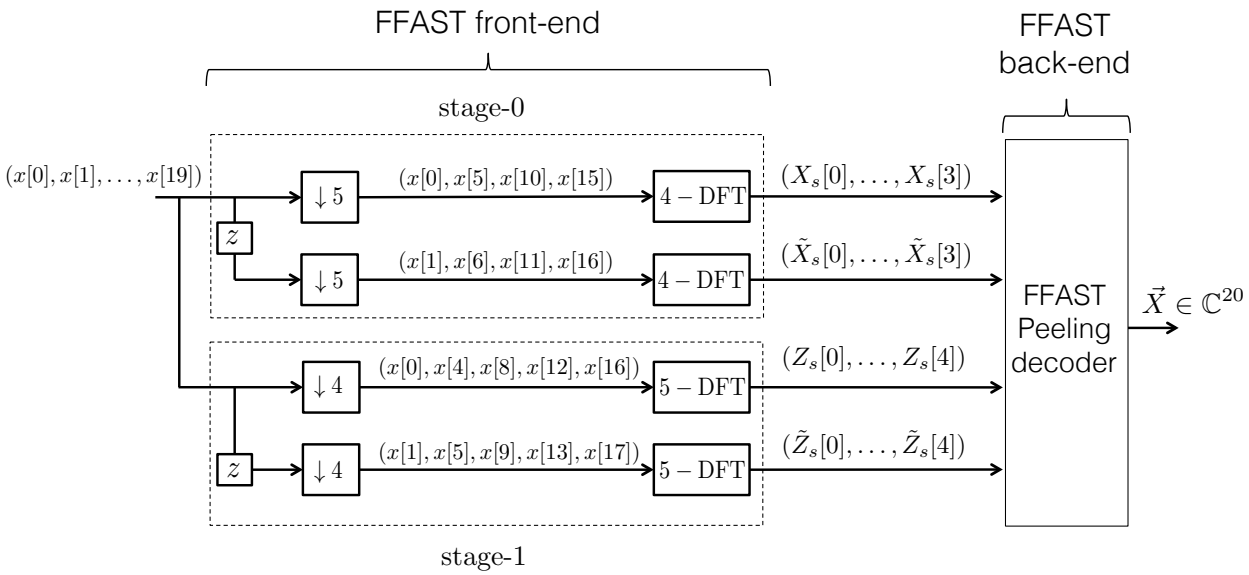


Figure 6.1: A toy-example of the FFAST architecture. The input to the FFAST architecture is a 20-point discrete-time signal $\vec{x} = (x[0], \ldots, x[19])$. The FFAST 'front-end', first samples the input signal and its circularly shifted version by 5 to obtain two sub-streams, each of length 4. A 4-point DFT of each sub-stream is then computed to obtain the observations $(X_s[.], \tilde{X}_s[.])$. Similarly, downsampling by 4 followed by a 5-point DFT provides the second set of 5 observations $(Z_s[.], \tilde{Z}_s[.])$. The FFAST 'back-end' consists of a peeling-decoder that synthesizes the big 20-point DFT $\vec{X}$, from the short DFT's of each of the sub-sampled data stream. In general, the sub-sampling front-end of the FFAST architecture consists of 3 or more stages depending on the sparsity index δ, where $k = O(n^{\delta})$. In this example, we show the FFAST architecture with 2-stages only for the purpose of illustration.

of bin 1 in stage 0 and is given by,

$$\vec{y}_{b,0,1} = \begin{pmatrix} X_s[1] \\ \tilde{X}_s[1] \end{pmatrix}. \tag{6.10}$$

The first index of the observation vector corresponds to the stage number, while the second index is the bin number within a stage. Note, that in the FFAST architecture of Fig. 6.1, there are total of 4 bins in stage 0 and 5 bins in stage 1.

The relation between the bin-observations and the non-zero DFT coefficients $\vec{X}$ can be computed using the signal processing properties of sub-sampling and circular shift. A graphical representation of this relation is shown in Fig. 6.2. Left nodes of the graph in Fig. 6.2 represent the non-zero DFT coefficients and the right nodes represent the "bins" (check nodes) with vector observations. An edge connects a left node to a right check node iff the corresponding non-zero DFT coefficient contributes to the observation vector of that particular check node, e.g., after aliasing, due to sub-sampling, the DFT coefficient $X[10]$ contributes to the observation vector of bin 2 of stage 0 and bin 0 of stage 1.

We define the following:

- **zero-ton**: A bin that has no contribution from any of the non-zero DFT coefficients

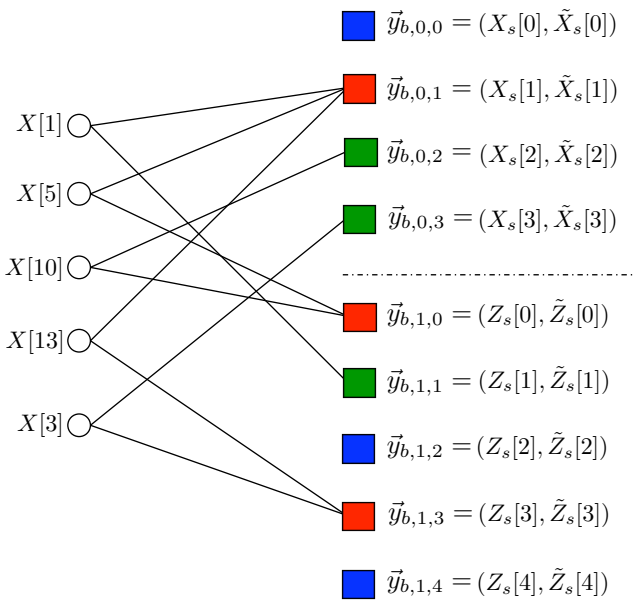


Figure 6.2: A 2-left regular degree bi-partite graph representing the relation between the unknown non-zero DFT coefficients and the observations obtained through the FFAST architecture shown in Fig. 6.1, for the 20-point example signal $\vec{x}$. Variable (left) nodes correspond to the non-zero DFT coefficients and the check (right) nodes are the observations. The observation at each check node is a 2-dimensional complex-valued vector e.g., $\vec{y}_{b,0,0} = (X_s[0], \tilde{X}_s[0])$.

of the signal, e.g., bin 0 of stage 0 or bin 2 of stage 1, as shown in Fig. 6.2. A zero-ton bin can be trivially identified from its observations.

- **single-ton**: A bin that has contribution from exactly one non-zero DFT coefficient of the signal, e.g., bin 2 of stage 0. Using the signal processing properties the observation vector of bin 2 of stage 0 is given as,

$$\vec{y}_{b,0,2} = \begin{pmatrix} X[10] \\ e^{2\pi\imath 10/20} X[10] \end{pmatrix}.$$

The observation vector of a singleton bin can be used to determine the support and the value, of the only non-zero DFT coefficient contributing to that bin, as follows:

  - *support*: The support of the non-zero DFT coefficient contributing to a singleton bin can be computed as,

$$10 = \frac{20}{2\pi} \angle \vec{y}_{b,0,2}[1] y^{\dagger}_{b,0,2}[0] \tag{6.11}$$

  - *Value*: The value of the non-zero DFT coefficient is given by the observation $y_{b,0,2}[0]$.

We refer to this procedure as a "ratio-test", in the sequel. Thus, a simple ratio-test on the observations of a singleton bin correctly identifies the support and the value of the only non-zero DFT coefficient connected to that bin. It is easy to verify that this property holds for all the singleton bins.

- **multi-ton**: A bin that has a contribution from more than one non-zero DFT coefficients of the signal, e.g., bin 1 of stage 0. The observation vector of bin 1 of stage 0 is,

$$\begin{aligned}\vec{y}_{b,0,1} &= X[1]\begin{pmatrix}1\\ e^{\imath 2\pi/20}\end{pmatrix} + X[5]\begin{pmatrix}1\\ e^{\imath 2\pi 5/20}\end{pmatrix} + X[13]\begin{pmatrix}1\\ e^{\imath 2\pi 13/20}\end{pmatrix}\\ &= \begin{pmatrix}10\\ -3.1634 - \imath 3.3541\end{pmatrix}\end{aligned}$$

Now, if we perform the "ratio-test" on these observations, we get, the support to be 12.59. Since, we know that the support has to be an integer value between 0 to 19, we conclude that the observations do not correspond to a singleton bin. In Section 2.A, we rigorously show that the ratio-test identifies a multi ton bin almost surely.

Hence, using the "ratio-test" on the bin-observations, the FFAST decoder can determine if a bin is a zero-ton, a single-ton or a multi-ton, almost surely. Also, when a bin is singleton the ratio-test provides the support as well as the value of the non-zero DFT coefficient connected to that bin. We use the following peeling-decoder on the graph in Fig. 6.2, to compute the support and the values of the non-zero DFT coefficients of $\vec{x}$.

FFAST peeling-decoder: The FFAST decoder repeats the following steps:

1. Select all the edges in the graph with right degree 1 (edges connected to single-tons).
2. Remove these edges from the graph as well as the associated left and right nodes.
3. Remove all the other edges that were connected to the left nodes removed in step-2. When a neighboring edge of any right node is removed, its contribution is subtracted from that check node.

Decoding is successful if, at the end, all the edges have been removed from the graph. It is easy to verify that performing the peeling procedure on the example graph of Fig. 6.2 results in successful decoding, with the coefficients being uncovered in the following possible order: $X[10], X[3], X[1], X[5], X[13]$.

Thus, the FFAST architecture computes the DFT of $\vec{x}$ by performing peeling-decoding over an appropriate bi-partite graph. Clearly the success the FFAST decoder depends on the properties of the sparse bi-partite graph resulting from the sub-sampling operation of the FFAST front-end. In Chapter 3, we provide a detailed description of how to judiciously choose the FFAST front-end parameters, depending on the problem dimensions (k, n), and

show using rigorous analysis that the FFAST peeling-decoder successfully computes a k-sparse n-length DFT, with high probability, using $O(k)$ samples and $O(k \log k)$ complex operations.

6.6 Univariate sparse polynomial interpolation over the complex field

In this section, we are interested in interpolating the non-zero coefficients of a k-sparse univariate polynomial f, with degree less than n, for the case where the coefficients $\{c_i\} \in \mathbb{C}$. We use the mapping of the univariate sparse polynomial interpolation problem to that of computing a sparse DFT as described in Section 6.5.2, and the FFAST architecture of Section 6.5.3, to construct the FFAST-based sparse polynomial interpolation algorithm over the complex field. The following theorem precisely characterizes the performance guarantees of the FFAST-based sparse polynomial interpolation algorithm.

Theorem 6.6.1. *For any given $\varepsilon > 0$, there exist (infinitely many) sufficiently large n, such that the FFAST-based sparse polynomial interpolation algorithm computes the non-zero coefficients of a k-sparse univariate polynomial, over the complex field, of degree less than n, where $k = \Omega(n^\delta)$ and $0 < \delta < 1$, with the following properties:*

1. **Sample complexity:** *The algorithm needs $m = r(\delta)k$ evaluations of the underlying polynomial, where $r(\delta) > 1$, is a small constant that depends on δ. For example, for $0 < \delta < 0.99$, the constant $r(\delta) < 4$.*

2. **Computational complexity:** *The computational complexity of the interpolation algorithm is*
   $O(k \log k)$, where the constant in big-Oh is small.

3. **Probability of success:** *The interpolation algorithm successfully recovers all the non-zero coefficients with probability at least $1 - \varepsilon$.*

Proof. We use the evaluations of the underlying sparse polynomial at specific powers of $\alpha = e^{\imath 2\pi/n}$, as dictated by the FFAST architecture. Then, the proof of the theorem follows from the mapping described in Section 6.5.2, and the results of Chapter 3. □

The algorithm of Ben Or and Tiwari [2] is based on *finding roots of polynomials*. The location of the roots can be very sensitive to the perturbations in the coefficients of the polynomial, i.e., is well-known to be a very *ill-conditioned* numerically. In contrast, the FFAST architecture naturally generalizes to a noise robust version, albeit at the expense of more sample and computational complexity. In Chapter 4, we provide a noise robust version of the FFAST algorithm, that can further be used to design a noise robust sparse polynomial interpolation algorithm over the complex field.

6.7 Univariate sparse polynomial interpolation over finite field

In this section, we discuss the adaptation of the FFAST algorithm to a sparse univariate polynomial interpolation over finite fields and the conditions under which this adaptation is applicable.

6.7.1 Finite field preliminaries

We recall some basic definitions related to finite fields. Consider a finite field $\mathbb{F}_q$ of size q. Then, we have the following:

Definition 6.7.1. *[Order of an element] An order of an element $\alpha \in \mathbb{F}_q$ is defined as the smallest positive integer $r \geq 1$, such that $\alpha^r = 1$. The order of all the non-zero elements $\alpha \in \mathbb{F}_q$, divides $(q-1)$, i.e., $r \mid (q-1)$.*

Definition 6.7.2. *[Discrete-Logarithm] Consider elements $\alpha, \beta \in \mathbb{F}_q$, such that $\beta = \alpha^i$, for some $i = 0, \ldots, q-2$. Then, the exponent 'i' can be computed using a* discrete-logarithm *of β with respect to base α, i.e.,*

$$i = \log_\alpha \beta.$$

Definition 6.7.3. *[DFT] Let $\alpha \in \mathbb{F}_q$, be a primitive n^{th} root of unity, i.e., the order of α is n. Then, one can define the following transform and its inverse,*

$$\begin{aligned} X[i] &= \sum_{j=0}^{n-1} x[j]\alpha^{-ij} \\ x[j] &= \sum_{i=0}^{n-1} X[i]\alpha^{ij}, \end{aligned}$$

where $\alpha \cdot \alpha^{-1} = 1$. The transform and its inverse as defined above satisfy all the basic discrete-Fourier-transform properties [4], e.g., sampling, aliasing, convolution etc.

6.7.2 FFAST-based sparse univariate polynomial interpolation algorithm for finite fields

The sparse univariate interpolation problem over a finite field $\mathbb{F}_q$ is given as follows. Let $f(x) \in \mathbb{F}_q[x]$ be a k-sparse univariate polynomial in x, defined over the field $\mathbb{F}_q$, with degree less than n, i.e.,

$$f(x) = \sum_{i=0}^{n-1} c_i x^i, \tag{6.12}$$

where at most k of the coefficients $c_i \in \mathbb{F}_q$ are non-zero. The polynomial interpolation problem is one of determining all the k non-zero coefficients of the polynomial f, from a set of evaluations of $f(x)$ at some elements of the finite field $\mathbb{F}_q$. Using techniques similar to those described in Section 6.6, for interpolating a sparse polynomial over the complex field, we propose a sparse polynomial interpolation algorithm for polynomials over finite fields. The proposed polynomial interpolation algorithm is applicable whenever the finite field $\mathbb{F}_q$ satisfies certain (mild and expected) conditions. This is not unlike the widely used *Reed-Solomon codes*, that require the field size to be larger than the block length of the code [51]. Next, we define two properties of a finite field $\mathbb{F}_q$, which are crucial for the FFAST-based sparse polynomial interpolation algorithm to exist.

Property 6.7.4. *The finite field $\mathbb{F}_q$ has a primitive n^{th} root of unity, where n is a positive integer greater than or equal to the highest degree of the polynomial f.*

The property 6.7.4 is required to define an n-length DFT, as in 6.7.3, over the finite field $\mathbb{F}_q$.

Property 6.7.5. *The finite field $\mathbb{F}_q$ is such that $(q-1)$ factors into "small" primes, i.e., the largest prime in the factorization of $(q-1)$ is much smaller than q (technically $o(q)$).*

In general, computing the discrete-logarithm over a finite field $\mathbb{F}_q$ is considered to be computationally hard, and hence finds applications in many cryptographic systems. The FFAST peeling-decoder needs to compute a discrete-logarithm while performing the 'ratio-test' in (6.11) over the elements from finite fields. In [63], the authors show that when $\mathbb{F}_q$ satisfies property 6.7.5 a discrete-logarithm can be computed efficiently using $O(\log^2 q)$ computations.

Consider a finite field $\mathbb{F}_q$ that satisfies the properties 6.7.4 and 6.7.5. Then, using the mapping of the univariate sparse polynomial interpolation problem to that of computing a sparse DFT described in Section 6.5.2, and the FFAST architecture of Section 6.5.3, we can construct the FFAST-based algorithm to interpolate a sparse polynomial over the field $\mathbb{F}_q$. The following theorem precisely characterizes the performance guarantees of the FFAST-based sparse polynomial interpolation algorithm.

Theorem 6.7.6. *For any given $\varepsilon > 0$, there exist (infinitely many) sufficiently large n, such that the FFAST-based sparse polynomial interpolation algorithm computes all the non-zero coefficients of a k-sparse univariate polynomial, over the finite field $\mathbb{F}_q$ that satisfies the properties 6.7.4 and 6.7.5, of degree less than n, where $k = \Omega(n^\delta)$ and $0 < \delta < 1$, with the following properties:*

1. **Sample complexity:** *The interpolation algorithm needs $m = r(\delta)k$ evaluations of the underlying polynomial, where $r(\delta) > 1$ is a small constant that depends on δ. For example, for $0 < \delta < 0.99$, the constant $r(\delta) < 6$.*

2. **Computational complexity:** *The computational complexity of the interpolation algorithm is* $O(m\log(m)\log^2 q)$*, where the constant in big-Oh is small.*

3. **Probability of success:** *The interpolation algorithm successfully recovers the non-zero coefficients with probability at least* $1-\varepsilon$*.*

Proof. Please see Appendix 6.A □

Next, we provide some example constructions of the FFAST-based sparse polynomial interpolation algorithm, over finite fields.

Example 6.7.7. *Consider a finite field* $\mathbb{F}_q$ *of size* $q = 21601 = 2^5 3^3 5^2 + 1$*, where* q *is prime and let* $n = q-1 = 21600$*. Note, that the field* $\mathbb{F}_q$ *satisfies both the properties 6.7.4 and 6.7.5. We construct a* 3*-stage FFAST architecture with the sampling factors* $3^3 5^2$*,* $2^5 5^2$ *and* $3^3 2^5$ *for each of the* 3 *stages respectively. Then, using Theorem 6.7.6, we conclude that the resulting FFAST-based sparse polynomial interpolation algorithm perfectly recovers a* k *sparse* n*-degree polynomial over the field* $\mathbb{F}_q$*, using* $O(k)$ *evaluations and* $O(k\log k\log^2 q)$ *computations, with high probability.*

Example 6.7.8. *Consider a finite field* $\mathbb{F}_q$ *of size* $q = 5184001 = 8(2^6 3^4 5^3) + 1$*, where* q *is prime and let* $n = (q-1)/8 = 648000$*. Note, that the field* $\mathbb{F}_q$ *satisfies both the properties 6.7.4 and 6.7.5. We construct a* 3*-stage FFAST architecture with the sampling factors* $3^4 5^3$*,* $2^6 5^3$ *and* $3^4 2^6$ *for each of the* 3 *stages respectively. Then, using Theorem 6.7.6, we conclude that the resulting FFAST-based sparse polynomial interpolation algorithm perfectly recovers a* k *sparse* n*-degree polynomial over the field* $\mathbb{F}_q$*, using* $O(k)$ *evaluations and* $O(k\log k\log^2 q)$ *computations, with high probability.*

6.A Analysis of the sparse polynomial interpolation algorithm over finite fields

Consider a finite field $\mathbb{F}_q$ that satisfies the properties 6.7.4 and 6.7.5. Let $\alpha \in \mathbb{F}_q$, be a primitive n^{th} root of unity, i.e., $\alpha^n = 1$. We define the DFT over the field $\mathbb{F}_q$ using α, as in 6.7.3. Then, from [4] we know that the DFT over the finite field $\mathbb{F}_q$ satisfies all the sampling-aliasing and circular shift properties. Hence, processing an n-length discrete-time signal (corresponding to the evaluations of the underlying sparse polynomial at powers of α) with entries from the finite field $\mathbb{F}_q$, whose finite field DFT is k-sparse, through the FFAST sub-sampling front-end will result in a sparse bi-partite graph (see Fig. 6.2 for an example). The FFAST peeling-decoder can then recover all the non-zero DFT (or polynomial) coefficients, provided that it can identify the zero-ton, single-ton and multi-ton bins with high probability.

The FFAST architecture for the finite field case is similar to the one shown in Fig. 6.1, except that each stage has 3 sub-sampling paths instead of 2 as in Fig. 6.1. In other words,

the FFAST sub-sampling front-end samples the input signal and its 2 circularly shifted, i.e., z, z^2, versions to get 3 subsampled streams per stage. The output of the FFAST sub-sampling front-end is then obtained by computing short finite field DFT's of the 3 sub-sampled data streams. Similar to Section 6.5.3, we form the bin-observations as follows:

- **zero-ton**: A bin that has no contribution from any of the non-zero DFT coefficients of the signal. A zero-ton bin can be trivially identified from its observations.
- **single-ton**: A bin that has contribution from exactly one non-zero DFT coefficient of the signal. Then, using the sub-sampling and circular shift properties the bin-observation vector $\vec{y_b}$ of a singleton bin is given as,

$$\vec{y_b} = \begin{pmatrix} c_\ell \\ c_\ell \alpha^{\ell} \\ c_\ell \alpha^{2\ell} \end{pmatrix},$$

  where c_ℓ is the value of the only non-zero coefficient connected to this bin and ℓ is its support location. The observation vector of a singleton bin can be used to determine the support and the value of the only non-zero DFT coefficient contributing to that bin, as follows:

  - *support*: The support of the non-zero DFT coefficient contributing to a singleton bin can be computed as,

$$\begin{aligned} \text{if } \; (\log_\alpha y_b[1] y_b^{-1}[0] \; &== \; \log_\alpha y_b[2] y_b^{-1}[1]) \\ \text{then } \; \ell \; &= \; \log_\alpha y_b[1] y_b^{-1}[0] \end{aligned} \tag{6.13}$$

  - *Value*: The value of the non-zero DFT coefficient is given by the observation $y_b[0]$.

  Thus, a simple ratio-test on the observations of a singleton bin correctly identifies the support and the value of the only non-zero DFT coefficient connected to that bin. It is easy to verify that this property holds for all the singleton bins.

- **multi-ton**: A bin that has a contribution from $L > 1$ non-zero DFT coefficients of the signal. Then, using the sub-sampling and circular shift properties the bin-observation vector $\vec{y_b}$ of a singleton bin is given as,

$$\vec{y_b} = \begin{pmatrix} \sum_{i=0}^{L-1} c_{\ell_i} \\ \sum_{i=0}^{L-1} c_{\ell_i} \alpha^{\ell_i} \\ \sum_{i=0}^{L-1} c_{\ell_i} \alpha^{2\ell_i} \end{pmatrix},$$

  where $\{c_{\ell_i}, \ell_i\}_{i=0}^{L-1}$ are the values and the supports of the L non-zero coefficient connected to this bin. Later, we show that, with high probability, the "ratio-test" on these observations fail to satisfy the equality condition in (6.13), thus identifying a multiton bin.

Next, we compute the probability of the event that the ratio-test fails to identify a multiton bin.

6.A.1 Analysis of the ratio-test for a multiton bin

First consider a multiton bin with $L = 2$. Let, if possible the ratio-test on the multiton bin-observation, provide the value and the support of the non-zero coefficient to be c_{ℓ_2} and ℓ_2 respectively. Then, we have,

$$\vec{y}_b = \begin{pmatrix} 1 & 1 \\ \alpha^{\ell_0} & \alpha^{\ell_1} \\ \alpha^{2\ell_0} & \alpha^{2\ell_1} \end{pmatrix} \begin{pmatrix} c_{\ell_0} \\ c_{\ell_1} \end{pmatrix} = \begin{pmatrix} 1 \\ \alpha^{\ell_2} \\ \alpha^{2\ell_2} \end{pmatrix} c_{\ell_2}. \tag{6.14}$$

This is a *contradiction* since the matrix has a Vandermonde structure and hence invertible.

Now, consider the case where a multiton bin consists of $L > 2$ components. Then, it is easy to see that a uniformly random choice of the values of the coefficients $\{c_i\}$, induces a uniformly random distribution on the bin-observation $\vec{y}_b \in \mathbb{F}_q^3$. There are total of n possibilities for the support and q possibilities for the value of a singleton. Thus, the probability that a uniformly random vector in $\mathbb{F}_q^3$ is identical to some singleton is upper bounded by $n/q^2 < 1/n$.

Let E_b, be an event that the FFAST peeling-decoder makes an error in processing *any bin* during the whole decoding process. Then, applying union bound over $O(k)$ bins and constant iterations, we get, $Pr(E_b) < O(k/n)$. Now, if E_f denote the event that the FFAST decoder fails to recover all the non-zero coefficients. Then,

$$\begin{aligned} Pr(E_f) &< Pr(E_f \mid \overline{E_b}) + Pr(E_b) \\ &\overset{(a)}{<} O(1/k) + O(n/k) \\ &< \varepsilon. \end{aligned}$$

The inequality (a) follows from Chapter 3.

The computational complexity of the FFAST algorithm *without the discrete-log operation* is $O(k \log k)$. In [63], the authors have shown that, when the $\mathbb{F}_q$ satisfies property 6.7.5, then a discrete-logarithm can be computed efficiently using $O(\log^2 q)$ computations. Hence, the overall complexity of the FFAST-based sparse polynomial interpolation algorithm, over the finite field $\mathbb{F}_q$, is $O(k \log k \log^2 q)$. ■

Chapter 7

Conclusion and Future Research Directions

7.1 Conclusion

In this thesis, we considered the problem of computing a sparse Discrete-Fourier-Transform of a high-dimensional signal from its time-domain samples, as a representative example of compressed-sensing problems. Further, we used the problem of computing a sparse DFT to investigate the *tradeoff between the number of measurements, noise robustness and the computational complexity of the recovery algorithm, for a realistic sensing mechanism such as partial Fourier measurements*.

We proposed a new family of *deterministic sparse sensing matrices*, obtained by intellectually blending together diverse ideas from sparse graph codes, digital signal processing and number theoretic concepts like the Chinese-remainder-theorem (CRT). The specific sparse structure of the proposed family of measurement matrices further enables a Peeling-based Ultra-Low complexity algorithms for Sparse signal Estimation, that are accordingly dubbed as PULSE algorithms. The key idea is to perform a filterless subsampling of the input signal using a small set of uniform subsampling patterns, guided by the CRT, to cleverly exploit the resulting aliasing artifacts. Further, using the CRT, we established an intimate connection between the problem of computing a sparse DFT of a signal and decoding over an appropriately designed sparse graph code. This connection was then exploited 1) to design a *sample efficient* measurement matrix and a *low-complexity* peeling-style iterative recovery algorithm, and 2) to perform a rigorous analysis of the recovery algorithm by wielding powerful and well-established analytical tools such as *density-evolution*, *martingales*, and *expander graphs* from the coding theory literature. In particular, we have shown that under some mild conditions, a k-sparse n-length DFT of a signal can be computed using *nearly optimal* $4k$ measurements and $O(k \log k)$ computations. We also extended these results to the cases of noise-corrupted observations, computing sparse $2D$-DFTs as well as to interpolation of

multi-variate sparse polynomials over the complex field and finite fields.

To summarize, we augment the compressive-sensing literature by a *family of sample efficient deterministic sparse measurement matrices and the associated family of a low-complexity PULSE recovery algorithms*. However, there are a few caveats that we would like to emphasize. First, our results are for asymptotic regimes of the sparsity k and the signal dimension n, where k is sub-linear in n. Secondly, we assume a stochastic model on the input signal, in particular, we assume that the k-sparse signal has a uniformly random support. Thus, our proposed PULSE algorithms trades off the sample and the computational complexity for asymptotically zero probability of failure guarantees in a non-adversarial sparsity setting.

We have also implemented a C++ prototype of our FFAST (Fast Fourier Aliasing-based Sparse Transform) algorithm, which is a member of the family of PULSE algorithms. We use this prototype to empirically validate the theoretical claims of the performance of the proposed sparse sensing matrices and the FFAST algorithm. All the simulation results provided in this thesis were generated using this code. Implementation details of the code can be found at http://www.eecs.berkeley.edu/~kannanr/project_ffft.html

7.2 Future research directions

There are many interesting future research directions of this work. In the remainder of chapter, we attempt to categorize them into a few topics and provide a brief overview of each of them.

7.2.1 Modeling assumptions

In our problem formulation of computing a k-sparse n-length DFT of a signal from its time-domain samples, we have assumed a stochastic model for the input signal. In particular, we assume that the non-zero DFT coefficients of the n-length signal have a uniformly random support. For this stochastic model of the input signal, we have proposed a family of *deterministic* sparse measurement matrices and designed associated *deterministic* low-complexity recovery algorithms. The proposed algorithms thus, provide a *deterministic bound* on the sample and the computational complexity with *probabilistic recovery guarantees* of the input signal. Many real world applications have signals with a non-uniform spectral support, e.g., speech signals have most of its energy concentrated in low frequencies, while relatively less energy in higher frequencies. For such settings, one may need to study a systematic extension of the ideas and the results of this thesis to accommodate other stochastic models of the input signal. Additionally, some applications may require *worst-case or deterministic guarantees on the input signal* while being tolerant to *probabilistic guarantees of the sample and the computational complexity* of the recovery algorithm.

Non-uniform support

In Chapter 5, we demonstrated an application of the $2D$-FFAST framework to acquire the magnetic-resonance-image of the 'Brain' image shown in Fig. 7.1(a). Thus, providing an empirical evidence of our algorithm being applicable to signals with a "non-uniform" (or clustered) support for the dominant DFT coefficients, such as the 'Brain' image of Fig 5.8(b). A systematic study of extending the FFAST $1D$ and $2D$ frameworks for signals with a non-uniform spectral support would be of interest.

Worst-case input signal

We have proposed a family of *deterministic* sparse measurement matrices and designed associated *deterministic* low-complexity recovery algorithms. This provides a *deterministic bound* on the sample complexity and the computational complexity with *probabilistic recovery guarantees* of the input signal. We can turn around this approach, to provide *probabilistic bounds on the sample and the computational complexity* of the recovery algorithm, while *deterministically accommodating worst-case input signals*. For example, in the absence of observation noise, consider the following hybrid recovery scheme. First use the efficient FFAST algorithm to recover the input signal with a high probability. If the FFAST algorithm fails to recover all the non-zero coefficients, then use a computationally expensive algorithm like [64] or [2], that deterministically works for any support of the non-zero coefficients, on the residual signal. For this hybrid approach, one can show that the *expected* sample and computational complexity of recovering a k-sparse n-length signal, from linear noiseless observations, is $O(k)$ and $O(k \log k)$. The expectation is computed over the probability space of input signals with a uniformly random support. Such an approach provides a probabilistic bound on the sample complexity and the computational complexity of the recovery algorithm, while *deterministically* accommodating all the input signals.

7.2.2 Analysis

In this thesis we have analyzed the following two cases:

Noiseless

In Chapter 3, we show that the FFAST algorithm computes a k-sparse n-length DFT, using $O(k)$ time-domain samples of the input signal and $O(k \log k)$ computations. If we assume that the FFT algorithm is computationally optimal, i.e., $\Omega(N \log N)$ computations are necessary to compute an N-length DFT of an arbitrary N-length signal, then the FFAST algorithm is *order optimal* w.r.t both the sample complexity and the computational complexity.

Noisy

A standard tool used in the compressed sensing literature for the analysis of reconstruction algorithms is the *restricted isometry property* (RIP) [12]. The RIP essentially characterizes matrices which are nearly orthonormal or unitary when operating on sparse vectors. Although random measurement matrices like Gaussian matrices exhibit the RIP constants with optimal $O(k\log(n/k))$ scaling of the sample complexity, they have limited use in practice, and are not applicable to our problem of computing a sparse DFT from the time-domain samples. For a measurement matrix consisting of m rows from an $n \times n$ discrete Fourier transform matrix, slightly weaker estimates of RIP constants are available [8, 70, 65, 66]. For example, a stable recovery of a k-sparse n-length signal is guaranteed for the case when the number of measurements scale as $O(k\log^3 k\log n)$.

The analysis of the FFAST recovery algorithm provided in Chapter 4, also uses the RIP condition, to show a stable recovery of the signal. Specifically, we show that the FFAST algorithm recovers a k-sparse n-length DFT of a signal from $O(k\log^2 k\log n)$ noise-corrupted time-domain samples, with a high probability. The improvement in the sample complexity is obtained as a result of a divide-and-conquer approach of the FFAST algorithm, wherein we analyze the RIP constants of individual bin-level measurement matrices rather than the full measurement matrix. Thus, the divide-and-conquer approach of the FFAST architecture is not only useful in designing a low complexity recovery algorithm, but also provides a better bounds on the sample complexity. Unfortunately, the analysis of the FFAST algorithm, specifically of the "multi-ton" bins, is not tight. The simulation results in Chapter 4, provide ample empirical evidence that the partial Fourier measurements exhibit nearly-optimal scaling of the number of measurements required for a stable recovery, i.e., $O(k\log n)$. The key component in tightening the multi-ton bin analysis is to use probabilistic bounds on the RIP constants rather than the worst-case bounds as used in Chapter 4.

7.2.3 Algorithms

The FFAST sub-sampling front-end divides the original problem of computing a k-sparse n-length DFT into multiple "bin-level" simpler problems (see Chapter 4 for details), using the CRT-guided small set of uniform subsampling patterns. Most of these bin-level sub-problems are trivial, of computing a 0-sparse DFT or 1-sparse DFT, while the others are almost trivial, i.e., of computing a $\log k$ sparse DFT. The FFAST decoder then performs the following iterative peeling procedure: it first identifies an instance of a sub-problem that is 1-sparse and reliably computes the support and the value of the non-zero DFT coefficient participating in this sub-problem. Then, it *peels off* the contribution of the identified non-zero DFT coefficient from other sub-problems, to create more instances of 1-sparse sub-problems. This peeling-style iterative recovery algorithm eventually uncovers all the non-zero DFT coefficients.

Bin-processing

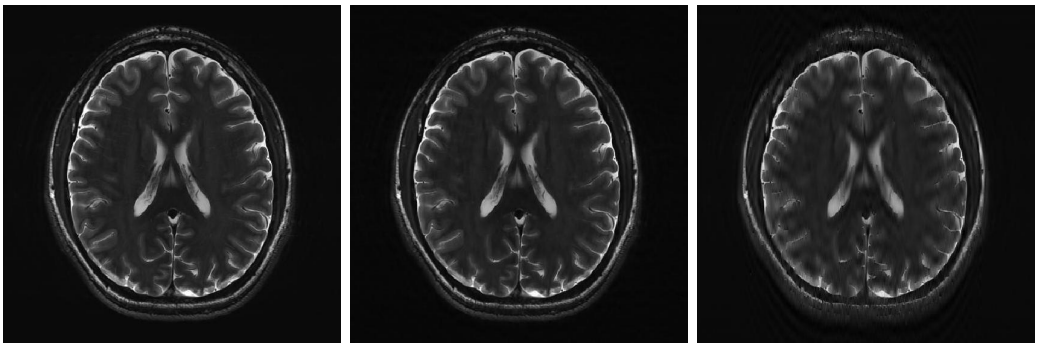
The FFAST decoder achieves the robustness against observation noise by carefully designing the bin-measurement matrices, and performing a robust bin-processing/reconstruction algorithm. The bin-processing algorithm (proposed in Section 4.6) performs an *exhaustive search* over n/k columns of the corresponding bin-measurement matrix, thus resulting in an overall computational complexity to be *super-linear* in the signal length n. A more systematic and *structured design of the bin-measurement matrices* is required to design a sub-linear time robust bin-processing algorithm.

Message passing algorithm

The FFAST peeling-style iterative decoder proposed in this thesis is based on the peeling-decoder used in the *erasure correcting sparse graph codes* [53]. A key property in decoding for the erasures is that the messages flowing between the parity-check nodes and the variable nodes of the sparse graph are *error-free*. However, in the presence of observation noise, the messages from the check nodes to the variable nodes in the FFAST decoder have an asymptotically small (but non-zero) probability of being erroneous. More sophisticated techniques like "belief-propagation", if used appropriately, in the FFAST decoder, may provide better results in terms of the robustness against observation noise.

Iterative-soft-thresholding (IST) back-end algorithm

The FFAST architecture consists of a sub-sampling "front-end" and a peeling-decoder "back-end" (for example, see Fig. 4.2). The FFAST front-end structure induces the sparse *measurement matrix* $\boldsymbol{A}$. In Chapter 4, we have established the RIP and mutual incoherence properties of $\boldsymbol{A}$ and its sub-matrices (bin-measurement matrices). In principle, one can use the FFAST front-end, i.e., $\boldsymbol{A}$, in conjunction with the well studied reconstruction algorithms from compressed-sensing literature, e.g., LASSO (least absolute shrinkage and selection operator), IST, etc. As an example, consider a $2D$-FFAST sub-sampling front-end that acquires 60.18% of the Fourier samples of the 'Brain' image shown in Fig. 7.1(a). We can reconstruct the Brain image from these samples using either an IST back-end algorithm, as shown in Fig. 7.1(b), or the FFAST peeling-decoder, as shown in Fig. 7.1(c). We note that for the same input Fourier samples, the IST back-end performs a higher quality recovery as compared to the FFAST peeling-decoder. For specific details of the simulation set-up, see Section 5.5.3 of Chapter 5. The FFAST sub-sampling front-end thus provides a new family of measurement matrices that can also be used in conjunction with the well studied reconstruction algorithms from the compressed-sensing literature. A formal analysis of such a combination of the FFAST front-end and IST back-end algorithms would be of interest.



(a) Original 'Brain' image of size 504×504 in spatial-domain.

(b) Reconstructed 'Brain' image using the $2D$-FFAST front-end and the IST back-end. The total number of Fourier samples used is 60.18%.

(c) Reconstructed 'Brain' image using the $2D$-FFAST front-end and the FFAST peeling back-end. The total number of Fourier samples used is 60.18%.

Figure 7.1: An application of the $2D$-FFAST sub-sampling front-end to acquire the Fourier samples of the 'Brain' image of size 504×504. The Brain image is then reconstructed from these samples using, b) an IST back-end algorithm, and c) the FFAST peeling back-end. In both the reconstructions we use 60.18% of the Fourier samples.

7.2.4 Sampling of continuous-time signals

In this thesis, we have explored the problem of recovering an n-length discrete-time signal, whose DFT is k-sparse, from its time-domain samples. The sub-sampling front-end of the proposed FFAST architecture consists of a few stages of a low-rate uniform sampling units. Thus, making the FFAST front-end amenable to hardware implementation. An extension of the FFAST architecture to sample a continuous-time signal would result in sub-Nyquist, low-power, wide-band analog-to-digital-converter (ADC) designs, that have numerous engineering applications. As a preliminary investigation of this topic, we empirically applied the FFAST architecture to acquire a continuous-time *discrete-multi-tone* (DMT) signals. We provide the results of this preliminary investigation in the sequel.

Discrete-Multi-Tone signals

A discrete-multi-tone signal $x(t)$ is a summation of a few, say k, complex sinusoids. The problem of acquisition of the continuous-time signal $x(t)$ can be discretized as follows. Let $f_{\max}$ be the highest frequency in the signal $x(t)$. First, sample the continuous-time signal $x(t)$ at the Nyquist-rate of $2f_{\max}$, for a duration T, to get $n = 2Tf_{\max}$ samples. Then, compute the n-point DFT of the sub-sampled signal. If the sinusoidal components of the signal $x(t)$ are *harmonically related* with a fundamental frequency of $1/T$, then the DFT of the sub-sampled signal has precisely k non-zero terms and the signal $x(t)$ can be reconstructed perfectly from the DFT. However, if the sinusoidal components of the signal $x(t)$ are not harmonically related, then the DFT of the sub-sampled signal has k dominant terms with

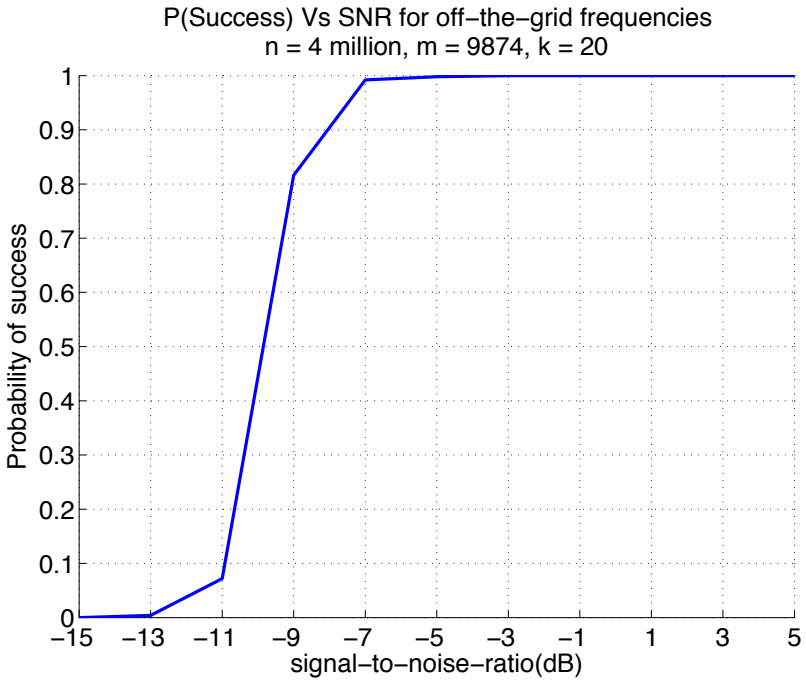


Figure 7.2: A plot of the probability of success of the FFAST algorithm for various values of the signal-to-noise ratio. A continuous-time DMT signal $x(t)$ of duration $T = 2$ seconds, with $k = 20$ sinusoidal components is generated. The unknown frequencies are real valued between -1MHz and 1MHz. The Nyquist-rate sampling generates $n = 4$ million samples in a 2 seconds duration. The FFAST architecture reconstructs the signal $x(t)$ from $m \approx 10,000$ noise-corrupted samples, using the "Blackman-Nuttall" window to reduce the Gibbs-effect. A successful run corresponds to the case where the FFAST algorithm recovers all the $k = 20$ sinusoidal components with frequency estimates within ± 2 Hz of the true frequency. Each point in the plot is obtained by averaging over 200 runs of the simulations.

many small terms due to the *Gibbs effect* that arises as a result of windowing. In compressed-sensing literature this problem is known as the "off-the-grid compressed sensing" problem [72]. In both the cases, i.e., on-the-grid or off-the-grid, the reconstructed signal $\hat{x}(t)$ obtained using the aforementioned discretization approach provides the frequency estimates with a small error, e.g., $|f_i - \hat{f}_i| < 1/n$, if there is a sufficient separation between any two frequency components, specifically if $|f_i - f_j| > O(1/n)$ for $i \neq j$. The sample complexity of this approach is n, and the resolution of the estimates is $O(1/n)$. We empirically observe that the FFAST architecture, provides similar resolution guarantees with merely $O(k \log n)$ samples. This can be a significant advantage in terms of designing low-rate ADC's.

For example, consider a signal $x(t)$ of duration $T = 2$ sec, with $k = 20$ sinusoidal components. Let the maximum frequency $f_{\max}$ be less than 1MHz. Then, using the Nyquist-rate sampling we get $n = 4$ million samples in 2 sec duration. The FFAST architecture reconstructs the signal $x(t)$ from $m \approx 10,000$ noise-corrupted samples, using the "Blackman-Nuttall" window to reduce the Gibbs-effect. In Fig. 7.2, we provide a plot of the probability of success of the FFAST algorithm for various values of the signal-to-noise ratio. A successful

run corresponds to the case where the FFAST algorithm recovers all the $k = 20$ sinusoidal components with frequency estimates within ± 2 Hz of the true frequency. Each point in the plot is obtained by averaging over 200 runs of simulations. The signal-to-noise ratio is defined as $\int_{t=0}^{T} |x_i(t)|^2 dt / \mathbb{E}\{\int_{t=0}^{T} |w(t)|^2 dt\}$ for each sinusoidal component $i = 0, 1, \ldots, 19$, where $w(t)$ is a white Gaussian noise process. A systematic study of the window design principles, as well as the analysis of the theoretical guarantees to acquire a continuous-time signal using the FFAST framework would be paramount interest.

Many applications like cognitive radio have signals with *sparse band occupancy* rather than *discrete-multi-tone*. The extension of the FFAST architecture to acquire multi-band signals with sparse band occupancy is related to the multiple measurement vector (MMV) problem [24, 58] in the compressed-sensing literature.

7.2.5 Extensions to other transforms

In this thesis, we have shown the application of the FFAST framework to compute the $1D$ and $2D$ DFTs of signals from the time-domain and the space-domain samples respectively. Recently, the authors of [37] extended the FFAST framework to compute a fast sparse Hadamard transform. It would be interesting to extend the FFAST framework to other transforms such as wavelet transform, discrete-cosine-transform (DCT), time-domain sparse signals, etc.

In summary, the proposed FFAST architecture provides a new and promising direction for acquiring signals that have a sparse structure in the transform-domain, e.g., the Fourier-domain. The framework and the recovery algorithm is sample efficient and robust against observation noise. There are many open challenges that need to be addressed and we have discussed some of them in earlier sections. Overall, we believe that the FFAST framework provides a possible direction towards designing sample efficient, low-power, engineering solutions for sparse signal acquisition.

Bibliography

[1] M. Bakshi, S. Jaggi, S. Cai, and M. Chen. Sho-fa: Robust compressive sensing with order-optimal complexity, measurements, and bits. *Arxiv preprint arXiv:1207.2335*, 2012.

[2] Michael Ben-Or. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309. ACM, 1988.

[3] R. Berinde, P. Indyk, and M. Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008.

[4] R.E. Blahut. *Fast algorithms for digital signal processing*. 1985.

[5] T. Blu, P.L. Dragotti, M. Vetterli, P. Marziliano, and L. Coulot. Sparse sampling of signal innovations. *Signal Processing Magazine, IEEE*, 25(2):31–40, 2008.

[6] John W Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM Computer Communication Review*, volume 28, pages 56–67. ACM, 1998.

[7] E Candes and J Romberg. l1-magic: Recovery of sparse signals via convex programming (2005). *URL: www.acm.caltech.edu/l1magic/downloads/l1magic.pdf*, 2006.

[8] E.J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[9] E.J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[10] E.J. Candes, J.K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8): 1207–1223, 2006.

[11] Emmanuel J Candes and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. *Foundations of Computational Mathematics*, 6 (2):227–254, 2006.

[12] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[13] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted 1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6): 877–905, 2008.

[14] Venkat Chandar, Devavrat Shah, and Gregory W Wornell. A simple message-passing algorithm for compressed sensing. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 1968–1972. IEEE, 2010.

[15] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002.

[16] Graham Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[17] Graham Cormode and S Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Structural Information and Communication Complexity*, pages 280–294. Springer, 2006.

[18] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. Technical report, DTIC Document, 2008.

[19] Geoffrey M Davis, Stephane G Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions. *Optical Engineering*, 33(7):2183–2191, 1994.

[20] D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4): 1289–1306, 2006.

[21] P.L. Dragotti, M. Vetterli, and T. Blu. Sampling moments and reconstructing signals of finite rate of innovation: Shannon meets strang–fix. *Signal Processing, IEEE Transactions on*, 55(5):1741–1757, 2007.

[22] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[23] Michael Elad, Boaz Matalon, Joseph Shtok, and Michael Zibulevsky. A wide-angle view at iterated shrinkage algorithms. In *Optical Engineering+ Applications*, pages 670102–670102. International Society for Optics and Photonics, 2007.

[24] Ping Feng and Yoram Bresler. Spectrum-blind minimum-rate sampling and reconstruction of multiband signals. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 3, pages 1688–1691. IEEE, 1996.

[25] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2007.

[26] M. Finiasz and K. Ramchandran. Private stream search at the same communication cost as a regular search: Role of ldpc codes. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 2556–2560. IEEE, 2012.

[27] Robert Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[28] Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27):2659–2662, 2009.

[29] Semyon Aranovich Gershgorin. Uber die abgrenzung der eigenwerte einer matrix. *Proceedings of the Russian Academy of Sciences. Mathematical series*, pages 749 – 754, 1931.

[30] Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, and Lixin Shi. Sample-optimal average-case sparse fourier transform in two dimensions. *arXiv preprint arXiv:1303.1209*, 2013.

[31] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 152–161, New York, NY, USA, 2002. ACM. ISBN 1-58113-495-9. doi: 10.1145/509907.509933. URL http://doi.acm.org/10.1145/509907.509933.

[32] A. C. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse fourier representations. In *in Proc. SPIE Wavelets XI*, 2003.

[33] AC Gilbert, MJ Strauss, JA Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *Arxiv preprint cs/0608079*, 2006.

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

[35] Anna C Gilbert, Martin J Strauss, and Joel A Tropp. A tutorial on fast fourier sampling. *Signal Processing Magazine, IEEE*, 2008.

[36] Irving John Good. The interaction algorithm and practical fourier analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 361–372, 1958.

[37] Saeid Haghighatshoar, Robin Scheibler, Martin Vetterli, et al. A fast hadamard transform for signals with sub-linear sparsity. In *51st Annual Allerton Conference on Communication, Control, and Computing*, 2013.

[38] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Nearly optimal sparse fourier transform. In *Proc. of the 44th SOTC*. ACM, 2012.

[39] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse fourier transform. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1183–1194. SIAM, 2012.

[40] Ling Huang, Jinzhu Jia, Bin Yu, Byung-Gon Chun, Petros Maniatis, and Mayur Naik. Predicting execution time of computer programs using sparse polynomial regression. In *Advances in Neural Information Processing Systems*, pages 883–891, 2010.

[41] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the l1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008.

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

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

[44] Sina Jafarpour, Weiyu Xu, Babak Hassibi, and Robert Calderbank. Efficient and robust compressed sensing using optimized expander graphs. *Information Theory, IEEE Transactions on*, 55(9):4299–4308, 2009.

[45] Seyed Mohammad Mahdi Javadi and Michael Monagan. Parallel sparse polynomial interpolation over finite fields. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, pages 160–168. ACM, 2010.

[46] Erich Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In *Symbolic and Algebraic Computation*, pages 467–474. Springer, 1989.

[47] Erich L Kaltofen. Fifteen years after dsc and wlss2 what parallel computations i do today: invited lecture at pasco 2010. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, pages 10–17. ACM, 2010.

[48] Vassilis Kekatos and Georgios B Giannakis. Sparse volterra and polynomial regression models: Recoverability and estimation. *Signal Processing, IEEE Transactions on*, 59 (12):5907–5920, 2011.

[49] M Amin Khajehnejad, Alexandros G Dimakis, Weiyu Xu, and Babak Hassibi. Sparse recovery of nonnegative signals with minimal expansion. *Signal Processing, IEEE Transactions on*, 59(1):196–208, 2011.

[50] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale¡ i¿ l¡/i¿¡ sub¿ 1¡/sub¿-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007.

[51] Shu Lin and Daniel J Costello. *Error control coding*, volume 123. Prentice-hall Englewood Cliffs, 2004.

[52] M.G. Luby, M. Mitzenmacher, and M.A. Shokrollahi. Analysis of random processes via and-or tree evaluation. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 364–373. Society for Industrial and Applied Mathematics, 1998.

[53] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Efficient erasure correcting codes. *Information Theory, IEEE Transactions on*, 47(2):569–584, 2001.

[54] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Improved low-density parity-check codes using irregular graphs. *Information Theory, IEEE Transactions on*, 47(2):585–598, 2001.

[55] Michael Luby. Digital fountain, inc. luby@ digitalfountain. com. 2002.

[56] Michael Luby and Michael Mitzenmacher. Verification codes: simple ldpc codes for large alphabets. In *Proc. 40th Annu. Allerton Conf.*, 2002.

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

[58] M. Mishali and Y.C. Eldar. From theory to practice: Sub-nyquist sampling of sparse wideband analog signals. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2): 375–391, 2010.

[59] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[60] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9(3):317–334, 2009.

[61] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[62] Vladilen F Pisarenko. The retrieval of harmonics from a covariance function. *Geophysical Journal of the Royal Astronomical Society*, 33(3):347–366, 1973.

[63] Stephen Pohlig and Martin Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. *Information Theory, IEEE Transactions on*, 24(1):106–110, 1978.

[64] R Prony. Essai experimental–,-. *J. de l'Ecole Polytechnique*, 1795.

[65] Holger Rauhut. Stability results for random sampling of sparse trigonometric polynomials. *Information Theory, IEEE Transactions on*, 54(12):5661–5670, 2008.

[66] Holger Rauhut, Justin Romberg, and Joel A Tropp. Restricted isometries for partial random circulant matrices. *Applied and Computational Harmonic Analysis*, 32(2):242–254, 2012.

[67] T.J. Richardson and R.L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on*, 47(2): 599–618, 2001.

[68] Daniel Steven Roche. *Efficient Computation with Sparse and Dense Polynomials*. PhD thesis, University of Waterloo, 2011.

[69] R. Roy and T. Kailath. Esprit-estimation of signal parameters via rotational invariance techniques. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(7): 984–995, 1989.

[70] Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[71] R. Schmidt. Multiple emitter location and signal parameter estimation. *Antennas and Propagation, IEEE Transactions on*, 1986.

[72] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. Compressed sensing off the grid. *arXiv preprint arXiv:1207.6053*, 2012.

[73] Clive Temperton. Self-sorting mixed-radix fast fourier transforms. *Journal of computational physics*, 52(1):1–23, 1983.

[74] L. H Thomas. Using a computer to solve problems in physics. *Applications of Digital Computers*, 1963.

[75] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[76] M. Vetterli, P. Marziliano, and T. Blu. Sampling signals with finite rate of innovation. *Signal Processing, IEEE Transactions on*, 2002.

[77] Wilhelm Werner. Polynomial interpolation: Lagrange versus newton. *Mathematics of computation*, pages 205–217, 1984.

[78] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on*, 57(7):2479–2493, 2009.

[79] Weiyu Xu and Babak Hassibi. Efficient compressive sensing with deterministic guarantees using expander graphs. In *Information Theory Workshop, 2007. ITW'07. IEEE*, pages 414–419. IEEE, 2007.

[80] Richard Zippel. *Probabilistic algorithms for sparse polynomials*. Springer, 1979.