Sparse Encoding of Signals through Structured Random Sampling

by

Praveen Kumar Yenduri

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2012

Doctoral Committee:

Professor Anna C. Gilbert, Chair
Professor Michael P. Flynn
Associate Professor Clayton Scott
Professor Jun Zhang

To my beloved family, who constantly shower me with love, faith and support, though we are thousands of miles apart on opposite sides of the planet.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Anna Gilbert, for her constant support and guidance throughout the past few years. She has been a great mentor and source of inspiration. Without her, this thesis would not have been possible. I am also grateful to my committee members and co-advisors: Professor Michael Flynn, Professor Jun Zhang and Professor Clayton Scott. Their expert technical advice and expectations have constantly motivated me to achieve success.

I would also like to thank my friend Jae Young Park for his support and encouragement. I appreciate his feedback on my proposal and thesis a lot. I would also like to express my gratitude to my friends and class-mates, Arun Padakandla, Madhu sudhan Reddy, Phani Motamarri, Supreet Jeloka, Kishan Kunduru, Jitendra Kochhar and many others who have made my stay pleasant and enjoyable. I am especially grateful to Janardhan and Jeenal Yandooru, my friends and family away from home, for their help and guidance in my time of need. They have been a constant source of kindness and have provided me the much needed love, fun and company during my lonely days in Michigan.

I will always cherish the moments I had with many wonderful people I met in Ann Arbor. Peren Ozturan and Jillian Ong require special mention in this regard. It has been a true blessing to have a friend like Isha Patel, who has cheered and sup-

ported me throughout my ups and downs. I am also thankful to Rajkumar, Sandhya, Sowmya; my friends back home in India, for their love and best wishes throughout the PhD journey.

I am lucky to be blessed with a wonderful family, my parents, Murali and Prasanna, my sister Pavani and niece Akshara. I could not have come this far without the inspiration and support of my family, especially my mother. I will always be grateful for her dedication and the sacrifices she made for us. Last but not least, I would like to thank God for his continued blessings.

Om Asato Maa Sadgamaya...

Tamaso Maa Jyotir-Gamaya...

Mrityor-Maa Amritam Gamaya...

Om Shanti Shanti Shantihee !!

Oh lord, lead us from unreality (of transitory existence) to the reality (of self).

Lead us from the darkness (of ignorance) to the light (of spiritual knowledge).

Lead us from the fear of death to the knowledge of immortality.

Let there be peace everywhere !!

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Sparse Encoding of Signals through Structured Random Sampling

by

Praveen Yenduri

Chair: Anna Gilbert

The novel paradigm of compressive sampling/sensing (CS), which aims to achieve simultaneous acquisition and compression of signals, has received significant research interest in recent years. CS has been widely applied in many areas and several novel algorithms have been developed over the past few years. However, practical implementation of CS systems remains somewhat limited. This is due to the limited scope of many algorithms in literature when it comes to the employed measurement architectures. In several CS techniques, a key problem is that physical constraints typically make it infeasible to actually implement many of the random projections described in the algorithms. Also, most methods focus only on discrete measurements of the signal, which is not always practicable. Therefore, innovative and practical sampling systems must be carefully designed to effectively exploit CS theory in practice. This work focuses on developing techniques that randomly sample in time,

that are also characterized by the presence of some structure in the sampling pattern. The structure is leveraged to enable a feasible implementation of acquisition hardware, while the randomness ensures recovery of sparse signals via greedy pursuit algorithms. In certain cases, the presence of a predefined structure in the sampling pattern can be further exploited to obtain other advantages such as reducing the run-time of reconstruction algorithms.

The main theme in the thesis is to develop algorithms that bridge the gap between theory and practice of structured random sampling. The work is motivated by several application problems where structured random sampling offers attractive solutions. One of the applications involves development of a low-power architecture for analog-to-digital conversion (ADC), that incorporates time-domain processing and random sampling techniques. Improving energy efficiency in both ways, the developed ADC occupies a unique position in the literature of compressive sensing ADCs.

Similar techniques in structured random sampling are employed to develop a novel low-rate neuron model which encodes information present in sensory stimuli at a rate that is proportional to the actual amount of information present in the signal rather than its duration. The developed neuron model demonstrated superior performance in terms of sparse encoding and recovery error when compared to the neurons proposed earlier in the literature.

Along with techniques borrowed from theoretical computer science, structured random sampling has been successfully employed in designing a novel, distributive,

spectrum sensing scheme for application in wide-band cognitive radios. Simulations show that the proposed scheme exhibits a performance similar to that of a Nyquist rate method, even with high noise and severe under-sampling. Additional structure in random sampling was further utilized to develop a sophisticated, resource-efficient, continuous sampling and reconstruction algorithm for quickly approximating the frequency content of spectrally-sparse digital signals.

CHAPTER I

Introduction

Compressive Sampling/Sensing (CS) is a novel sampling paradigm that exploits the redundancy present in many practical signals and images of interest to recover them from far fewer samples or measurements, typically well below the number required by the Shannon/Nyquist sampling theorem [2]. CS achieves this through two key ideas : (1) Sparsely representing the signals of interest in an appropriate basis and (2) Employing random measurements (signal projections) to extract maximum amount of information using only a minimum number of measurements. A key problem with many CS techniques in the literature is that physical constraints typically make it infeasible to actually measure many of the random projections described in the algorithms. Therefore, innovative and sophisticated sampling systems must be carefully designed to effectively exploit CS theory in practice. In this work, we focus on techniques that sample in time (which can be treated as linear projections of Fourier coefficients). We develop random sampling algorithms, that are also characterized by the presence of some structure in the sampling pattern. The structure is leveraged to enable a feasible implementation of acquisition hardware, while the randomness ensures recovery of sparse signals via greedy pursuit algorithms. We are motivated by several application problems where structured random sampling offers

attractive solutions. Our theme is to develop algorithms that bridge the gap between theory and practice of structured random sampling.

1.1 Compressive sensing basics

The basic idea of compressive sensing or compressive sampling is to exploit redundancy (i.e. sparsity or compressibility) in an input signal in order to reconstruct it from a small set of observations of the signal. In other words, compressive sensing aims for "smart" sampling of signals to acquire only the "important" information. In this way, the signal sampling rate can be reduced from the Nyquist rate to a rate that is proportional to the actual amount of information present in the input signal. The new sampling theory thus underlies procedures for sampling and compressing data simultaneously.

Let the signal of interest be represented by a vector x of length N. We say that x is sparse if it contains only a few non-zero components compared to the total length (N) of the signal. A compressible signal is one that is reasonably well approximated as a sparse signal. Let a vector $y = Ax$ represent linear measurements taken from x by the measurement system. The matrix, A, is called the measurement matrix and has a size K x N, where the number of measurements $K \ll N$. The reduction in the number of measurements that can be tolerated is proportional to the sparsity of the input signal x. The problem of recovering the signal x can be cast as that of solving an under-determined system of equations $Ax = y$. Solving for x based on y is an ill-posed problem in general, as there are infinitely many x that satisfy the relation $Ax = y$; however, it may be possible to uniquely solve for the input signal

x under the assumption that x is sparse or compressible. Of course, arbitrarily under-sampled linear measurements (i.e. arbitrary matrices A) will not succeed in recovering sparse vectors x. It has been shown that if the measurement matrix A satisfies the Restricted Isometric Property (RIP), then the sparse vector x can be recovered exactly [3]. A matrix is said to satisfy RIP with parameters (s, ϵ) for $\epsilon \in (0, 1)$, if for all s-sparse[1] vectors z,

$$(1 - \epsilon)||z||_2 \leq ||Az||_2 \leq (1 + \epsilon)||z||_2$$

Thus an RIP$(2s, \epsilon)$ matrix A approximately preserves the Euclidean length of $2s$-sparse vectors, which in turn implies that A approximately preserves the distance between any two s-sparse vectors. For example, if x_1 and x_2 are two s-sparse vectors, then $x_1 \neq x_2$ implies that $A(x1 - x2) \neq 0$. Thus the input x can be recovered by searching for the sparsest vector z that satisfies the condition, $Az = y$ or $||Az - y||_2 \leq \epsilon$ in case of noisy measurements. x can be expressed as the solution to the following optimization problem:

$$\arg\min ||z||_1 \quad \text{such that} \quad ||Az - y||_2 \leq \epsilon$$

There is no known algorithm that can verify if a given matrix is RIP other than the exponential time brute force algorithm. However various results have been published about the RIP nature of the matrix A if it is drawn from certain distributions of random matrices. For example, in the cases where A is a random Gaussian matrix [4], a random Bernoulli matrix [5] or a random partial DFT matrix [4], A satisfies RIP(s, ϵ) with high probability, if the number of measurements $K > O(\epsilon^{-2} s \log^{O(1)} N)$ (O(.) refers to the Big-O notation [3]). Algorithms that carry out the ℓ_1-minimization through linear programming to find x are typically referred to as the Basis Pursuit

[1] An s-sparse vector has at most s non-zero elements.

(BP) algorithms. BP algorithms are usually significantly slower (in theory) when compared to greedy pursuit algorithms ([6],[7],[8]), which limit the search space for x to s-sparse vectors, for a given s. Greedy pursuit algorithms try to minimize the ℓ_2-norm of the error (defined as $Ax - y$) subject to the condition that x is s-sparse:

$$\min||Ax - y||_2 \quad \text{such that} \quad ||x||_0 \leq s$$

Conventional greedy pursuit algorithms such as those proposed in [7] and [8], require the matrix A to be RIP.

The assumption of sparsity of x is justified by the fact that real world signals are often sparse or compressible in some transform domain. For example, communication signals such as FSK (frequency shift keying) are sparse in Fourier domain and natural images are often sparse in a Wavelet domain. In other words, even if the input signal x is not sparse, it can be represented as $x = WX$ for some sparse vector X, where W denotes the sparsifying transform matrix. The net measurement matrix now changes to $B = AW$ for the system $BX = y$. In this work, we are interested in input signals which are sparse in the frequency (Fourier) domain. In that case, X is the DFT of x and W is the IDFT (inverse discrete Fourier transform) matrix.

1.2 Structured random sampling

Most CS algorithms use RIP matrices whose entries are obtained independently from a standard probability distribution (e.g. random Gaussian measurements). However, such matrices are highly impractical and not feasible for real-world applications. In fact, very often the physics of the sensing modality and the capabilities of sensing devices limit the types of CS matrices that can be implemented in a specific

application. In contrast, in many applications, sampling in time can be efficiently implemented through the use of analog-to-digital converters. More over, sampling in time is the best option for frequency sparse signals, since time domain and frequency domain are maximally incoherent [9]. Hence, we restrict our attention to matrices that correspond to sampling in time. Now, it is also possible to construct deterministic sampling schemes that result in measurement matrices that satisfy the RIP property [10]. However, they require far more measurements when compared to random sampling schemes ($O(K^2)$ vs $O(K)$). Hence, we focus on random sampling. Applications, however, often do not allow the use of completely random matrices, but put certain physical constraints on the measurement process. This leads us to structured random sampling. The structure can be used to achieve a feasible implementation. In certain cases, the structure can also be used to obtain faster recovery algorithms. However imposing structure onto random sampling also has its disadvantages. The resultant measurement matrices do not necessarily satisfy the RIP condition, leading to the need to develop new algorithms and analysis. Depending on the structure imposed, there might also be other undesired consequences which have to be dealt with.

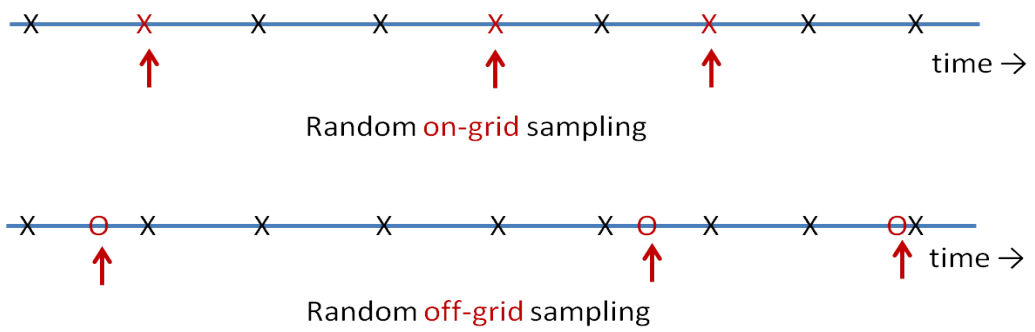

Figure 1.1: Figure showing the on-grid and off-grid sampling. The crosses represent the Nyquist grid.

Random sampling can be broadly classified into two categories, shown in Fig. 1.1. In on-grid random sampling the time points at which the signal is sampled are chosen randomly from a Nyquist grid (represented as crosses in the Fig. 1.1). On-grid random sampling can be viewed as random sub-sampling of a digital signal. Where as, in off-grid random sampling the time points have a continuous distribution and do not have to lie on the Nyquist-grid or any other grid. If the signal is observed during a time interval I, the off-grid time points at which the signal is sampled are continuous random variables with their range in I. The distribution of the random variables depends on the application. An example of an off-grid random sampling device is the random PPM ADC (presented in Chapter II). Another example is a level-crossing ADC, which samples the signal when it crosses some predefined amplitude levels.

Several fast sub-linear time algorithms can recover s-sparse signals from random on-grid samples ([11],[12],[13],[14]). These algorithms have a storage requirement and runtime of $O(s \log^{O(1)} N)$ (with the exception of [14], which samples the signal at an average rate close to Nyquist rate). A specific case of random off-grid sampling is studied in [15]. In this work, we develop both on-grid and off-grid structured random sampling techniques and investigate their application to different problems of interest.

1.3 Contributions

This thesis treats both theoretical and application aspects of structured random sampling. The main contributions of this thesis work, in structured random sampling,

are the following ([13],[16],[17],[18],[19],[20],[21]).

1.3.1 Theoretical

- **Periodic random sampling reconstruction (PRSreco) algorithm** (Section 2.7): We developed a new algorithm [16][20] for recovering frequency-sparse signals, from off-grid time samples, obtained in a periodically random pattern, at a sub-Nyquist rate. In periodic random sampling, the input signal is sampled at a random point within an interval of certain length and this process is repeated in every subsequent interval of that length. The algorithm is used for reconstruction in randomized time-based analog to digital converters that implement periodic random sampling. We analyze the algorithm and provide bounds on the reconstruction error. The PRSreco falls under the general category of greedy pursuit algorithms, but does not require the measurement matrix to be RIP. We also take a non-conventional approach in proving the error guarantees.

  Reference [16]: P.K. Yenduri, A.C. Gilbert, M.P. Flynn, and S. Naraghi, "Rand PPM: A low power compressive sampling analog to digital converter," *IEEE International Conf. on Acoustics, Speech and Sig. Processing (ICASSP)*, pp. 5980 − 5983, May 2011.

- **Continuous fast Fourier sampling (CFFS) algorithm** [13] (Chapter IV): Fourier sampling algorithms use a small number of structured random samples to quickly approximate the DFT of a spectrally-sparse digital signal from a given time window or block. Unfortunately, to obtain the spectral information on a particular block-of-interest, the samples acquired must be appropriately

structured for that block. Thus the sampling pattern forces a block-wise analysis and does not accommodate an arbitrary block analysis. We developed a new sampling procedure called Continuous Fast Fourier Sampling (CFFS) which samples the signal at sub-Nyquist rates and permits a sub-linear-time analysis of arbitrarily blocks of the signal. Thus, CFFS is a highly resource-efficient continuous sampling and reconstruction algorithm.

Reference [13]: P.K. Yenduri and A.C Gilbert, "Conitnuous fast fourier sampling," *In Proceedings of Sampling Theory and Applications (SAMPTA), Marseille, France*, 2009.

1.3.2 Applied

- **Random PPM: A low power compressive sampling time based ADC** [16] [20]: A random pulse-position-modulation (PPM) ADC architecture is proposed in Chapter II. A prototype 9-bit **random PPM** ADC incorporating a pseudo-random sampling scheme is implemented as proof of concept. This approach leverages the energy efficiency of time-based processing. The use of sampling techniques that exploit signal compressibility leads to further improvements in efficiency. The **random PPM** (pulse-position-modulation) ADC employs compressive sampling techniques to efficiently sample at sub-Nyquist rates. The sub-sampled signal is recovered using the PRSreco algorithm, which is tailored for practical hardware implementation. We develop a theoretical analysis of the hardware architecture and the reconstruction algorithm. Measurements of a prototype random PPM ADC and simulation, demonstrate this theory. The prototype successfully demonstrates a 90% reduction in sampling rate compared to the Nyquist rate for input signals that are 3% sparse in frequency domain.

Reference [20]: P.K. Yenduri, A. Rocca, A.S. Rao, S. Naraghi, A.C Gilbert, and M.P. Flynn, "A low power compressive sampling time-based analog to digital converter," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Special Issue on Circuits, Systems and Algorithms for Compressive Sensing*, Oct. 2012.

- **LowRate IAF: A sparse encoding model of neuron** [17] [18] (Chapter III): Neurons as Time Encoding Machines (TEMs) have been proposed to capture the information present in sensory stimuli and to encode it into spike trains. These neurons, however, produce spikes at firing rates above Nyquist rate, which is usually much higher than the amount of information actually present in stimuli. We propose a low-rate neuron which exploits the sparsity or compressibility present in natural signals to produce spikes at a firing rate proportional to the amount of information present in the signal rather than its duration, while using the spiking information in a smart manner to improve the performance of stimulus recovery.

  Reference [17]: P.K. Yenduri, A.C. Gilbert, and J. Zhang, "Model of a sparse encoding neuron," *Twenty First Annual Computational Neuroscience Meeting (CNS)*, Jul. 2012.

  Reference [18]: P.K. Yenduri, A.C. Gilbert, and J. Zhang, "Integrate-and-fire neuron modeled as a low-rate sparse time-encoding device," *Proceedings of Third International Conference on Intelligent Control and Information Processing (ICICIP)*, Jul. 2012.

- **Compressive and collaborative spectrum sensing for wide-band cognitive radios** [19] (Chapter V): One of the primary tasks of a cognitive radio (CR) is to monitor a wide spectrum and detect vacant channels, which can then be used for secondary transmission opportunities (i.e. for transmission by unlicensed users). CR systems thus enable dynamic spectrum access (DSA) and improve the overall efficiency of spectrum usage. However, the requirement of prohibitively high sampling rates to monitor a wideband, makes this a challenging task. In this work, we present a novel wideband spectrum sensing model that reduces the sampling requirement to a sub-Nyquist rate, proportional to the number of occupied channels in the wide spectrum. The sampling scheme is efficiently implementable using low-rate analog-to-digital converters (ADCs). The sensing algorithm uses techniques borrowed from theoretical computer science and compressive sampling, to detect the occupied channels with a high probability of success. The algorithm is implementable for spectrum sensing in a single CR, as well as in a decentralized CR-network with minimal communication between one-hop neighbors. The algorithm also has many other attractive features which make it different from other algorithms in literature.

  Reference [19]: P.K. Yenduri and A.C. Gilbert, "Compressive, collaborative spectrum sensing for wideband cognitive radios," *The Ninth International Symposium on Wireless Communication Systems (ISWCS)*, Aug. 2012.

CHAPTER II

Random PPM (Pulse Position Modulation) ADC

2.1 Introduction

Applications of low-power ADCs include power constrained wireless environmental sensing, high energy physics and biomedical applications such as massive-parallel access of neuron activity ([22],[23],[24]). We present a new low-power, compressive-sampling analog to digital converter which we call a random PPM ADC. The random PPM ADC is one of the first ADCs that takes advantage of the combination of time-based analog-to-digital conversion techniques and compressive sampling. In addition, we discuss a new reconstruction algorithm called PRSreco (Periodic Random Sampling reconstruction) and present theoretical upper-bounds for input signal reconstruction error. This algorithm is tailored to make it viable for practical hardware implementation.

Technology scaling generally improves power consumption and speed, however, it poses a number of challenges in the design of ADCs. Scaling reduces the supply voltage, which in turn reduces the signal dynamic range. This has the direct effect of reducing signal to noise ratio (SNR). One way to overcome the problems of low-voltage design is to process signals in time domain. Technology scaling favors time

domain processing since it reduces gate delays and thus improves time resolution. A wide variety of time-based ADCs that quantize time or frequency instead of voltage or current, have been proposed. These designs include simple architectures such as single-slope analog to digital conversion [25], pulse width modulation (PWM) ADC [26], asynchronous level crossing designs [27], VCO-based $\Sigma\Delta$ modulators [28] and integrate and fire circuits ([29],[30]). Continuous time DSPs are proposed in [31]. A continuous time level crossing ADC, such as [32] and [33] can be attractive for slow moving signals. However, the the key advantages of these devices are lost if continuous time DSP is not available. Furthermore, sparse signals can be dominated by high frequency content.

This work expands on the pulse position modulation ADC architecture developed in [1]. The PPM ADC is itself an elaboration of the PWM architecture in which a continuous-time comparator compares the input to a periodic ramp, to convert the input signal information to a time-domain representation (see Section 2.3). A two-step time-to-digital converter (TDC) then converts the time domain information to digital domain. With the use of a two-step TDC, the PPM ADC achieves both high resolution and high dynamic range, along with low power consumption. Another way to obtain an improvement in the power efficiency of an ADC is to reduce the sampling rate ([34]) since to a first order, power consumption is proportional to sampling frequency. We can achieve this by employing random sampling techniques that exploit the redundancy (i.e. compressibility or sparsity) of the input signal to reduce the sampling rates to below the Nyquist rate. We implement random sampling by introducing randomness into the reference ramp signal used by the PPM ADC. The proposed random PPM ADC lies at the intersection of time-based ADCs

and compressive-sampling ADCs, and thus improves efficiency in both ways.

Many compressive sampling (CS) ADC architectures and acquisition systems have been proposed in recent years. While some designs lack efficient implementation of CS encoding or decoding (reconstruction) algorithms in hardware ([35],[36]), other designs focus on efficient compression but not the optimization of power consumption ([37],[38]). Some compressive sensing designs, such as [39], employ a conventional high-speed ADC as an integral component. Also, none of the above designs use time-based conversion techniques to reduce power consumption. The random PPM ADC, thus occupies a unique position in the literature of compressive sensing ADCs.

The remainder of the chapter is organized as follows. Section 2.2 briefly relates the random sampling techniques used in this work with compressive sampling techniques that reduce the number of measurements needed to store and reconstruct a given input signal. The PPM and random PPM architectures are discussed in Section 2.3. A prototype random ADC, implemented as a custom CMOS PPM ADC coupled to an FPGA (Field-Programmable Gate Array) is described. The hardware implementation of the random PPM ADC is described in Section 2.4. The problem of reconstructing the input signal from ADC output samples is introduced in Section 2.5. In Section 2.7, we develop the PRSreco algorithm for the recovery of input signals that satisfy the signal model presented in Section 2.6. This new algorithm falls under the general category of greedy pursuit methods that aim to minimize the norm of the reconstruction error, subject to the sparsity conditions of the input signal. The PRSreco is analyzed in Section 2.7.1. The error bound of the recovered signal is discussed in Section 2.7.1. A second reconstruction algorithm called the

MOE (median of estimators) is developed and analyzed in Sec. 2.8. The appendices contain details about the mathematical modeling of the randomized sampling system along with lemmas and theorems that provide proof of correctness and run-time details of the algorithms.

The PRSreco algorithm and the MOE algorithm can also be used for signal reconstruction in other randomized time based ADCs as the analysis in Section 2.7.1 is easily extended. The algorithms are tailored to reduce computational cost and thus are viable for practical hardware implementation. Our analysis along with the numerical simulations and experimental results presented in Section 2.9, show that a random sampling time-based ADC exhibits much better performance than a non-random ADC operating at sub-Nyquist sampling rates.

2.2 Related Compressive Sampling (CS)

Consider the under-determined system of equations $BX = y$, where B is the net measurement matrix of size K x N, X is the DFT of the input signal vector x of length N and y is the measurement vector of length K ($K < N$). The sparse spectrum X can be expressed as the solution to the following optimization problem[1][40] :

$$\arg\min||X||_0 \quad \text{such that} \quad BX = y$$

The above problem requires the solution of a non-convex combinatorial problem, which is not practical [41]. Hence the ℓ_0-"norm" in the objective function is often

[1] $\arg\min||X||_0$ solves for X that has the smallest ℓ_0-"norm", where $||X||_0$ is defined as the number of non-zero elements in X.

replaced by its convex relaxation, the ℓ_1-norm[2]. That is,

$$\arg\min||X||_1 \quad \text{such that} \quad BX = y$$

It has been shown that if the measurement matrix B satisfies the Restricted Isometric Property (RIP), then the sparse vector X can be recovered exactly [3]. Algorithms that carry out the ℓ_1-minimization through linear programming to find X are typically referred to as the Basis Pursuit (BP) algorithms. Many of the proposed CS ADCs ([35],[36],[39]) use BP algorithms for reconstruction. However, BP algorithms are challenging to implement in hardware and are usually significantly slower when compared to greedy pursuit algorithms ([6],[7],[8]). Greedy pursuit algorithms try to minimize the ℓ_2-norm of the error (defined as $BX - y$) subject to the condition that X is s-sparse:

$$\min||BX - y||_2 \quad \text{such that} \quad ||X||_0 \leq s$$

Conventional greedy pursuit algorithms such as those proposed in [7] and [8], require the matrix B to be RIP. The measurement matrix B associated with the PPM ADC, does not necessarily satisfy the RIP condition. If a new matrix B is constructed randomly for each input signal X, then the RIP condition on B can be relaxed ([11],[6]). Hence, we impose the condition that B be a random matrix (newly constructed for each input signal X) and develop a new reconstruction algorithm, that falls under the category of greedy pursuit algorithms, but does not require matrix B to be RIP.

Different CS algorithms offer different error guarantees. We call the error guarantee ℓ_2/ℓ_1, if the following is true:

$$||X - \tilde{X}||_2 \leq \frac{C}{\sqrt{s}}||X - X_s||_1$$

[2]The ℓ_1-norm of a vector is defined as the sum of the absolute values of its elements.

where $\tilde{X}$ is the output of algorithm, C is a constant and X_s is the best s-term representation of X. A stronger error guarantee is the ℓ_2/ℓ_2, given by:

$$||X - \tilde{X}||_2 \leq C||X - X_s||_2$$

CS algorithms also offer different kinds of failure guarantees. Some methods fix the measurement matrix B and prove the reconstruction results for all input sparse signals X, while other algorithms can prove the reconstruction results with high probability for each input sparse signal X and a random measurement matrix B. Algorithms such as the BP [3], [7], [8] offer the stronger "for all" guarantee, but the weaker ℓ_2/ℓ_1 error guarantee. On the other hand Fourier sampling algorithms ([11],[12],[13]) offer stronger ℓ_2/ℓ_2 error guarantees and weaker "for each" failure guarantee. If a new measurement matrix B is randomly chosen for each sparse signal X, then a "for each" failure guarantee is sufficient. The output of an ADC is usually evaluated in terms of the reconstruction SNR, which involves the ratio of ℓ_2 norm of the signal to the ℓ_2 norm of the reconstruction error. Thus, a ℓ_2/ℓ_2 error guarantee is more suitable for such an analysis. The only way to obtain a ℓ_2/ℓ_2 error guarantee is to have a "for each" failure guarantee.

In this work, a "measurement" of the input signal is a measurement of the amplitude of the input signal at some time point. To obtain a random measurement matrix B, the input signal is sampled at random time points. The measurement vector y represents the amplitude of the signal at those random time points. Sub-linear time algorithms ([11],[12],[13],[14]) can recover s-sparse signals from random on-grid samples. However, PPM ADC produces off-grid samples. A specific case of random off-grid sampling is studied in [15] with a number of measurements, $K > O(sR^2\log(4N/\epsilon))$, where R is the dynamic range of X and ϵ is a tolerance parameter. In this chap-

ter, we deal with reconstruction from signal-dependent[3], random, off-grid samples. Thus the problem setting is different from [15], leading to an algorithm that offers different error guarantees and different conditions for recovery. We also take a non-conventional approach in proving the error guarantees. Unlike the random on-grid sampling techniques, our algorithm does not achieve a sub-linear run-time.

2.3 Hardware Design

In this section we describe the PPM ADC architecture and the design of the random PPM ADC design.

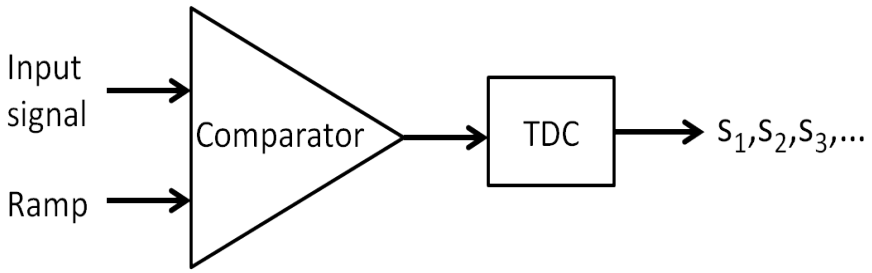

Figure 2.1: Block diagram of the PPM ADC

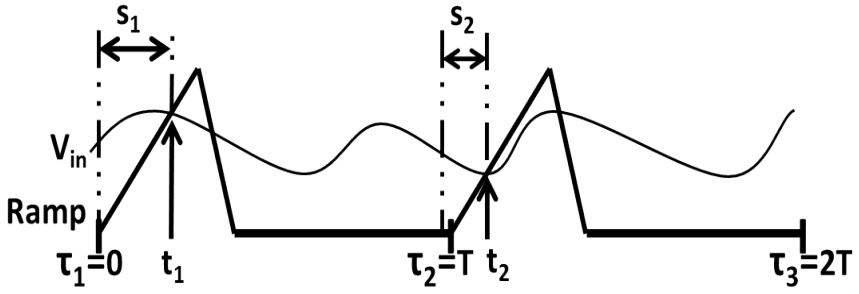

Figure 2.2: Waveforms depicting the sampling procedure in the PPM ADC

2.3.1 The PPM ADC Architecture

A block diagram of the PPM ADC is shown in Fig. 2.1. The sampling procedure [1] is depicted in Fig. 2.2. A comparator continuously compares the input signal

[3]The time points at which the signal is sampled depend on the signal, in contrast to being completely deterministic or completely random.

with a reference voltage ramp. An output pulse is generated by the comparator at the time instants where the ramp voltage exceeds the input signal. The time elapsed between the beginning of the ramp and the instant the input signal crosses the ramp (i.e. $s_1, s_2, ..$ as seen in Fig. 2.2) is measured and quantized by a two-step 9-bit time to digital converter (TDC). The simplest form of a TDC is a digital counter, however, to achieve a high resolution, one needs to have a very high counter frequency which in turn leads to a large energy consumption. On the other hand, delay line circuits [42] are more energy efficient for time measurement, however, the delay line must be long to measure long periods of time and can suffer from non-linearity. As a compromise, the two-step TDC consists of a 5-bit counter which performs coarse quantization and a delay line TDC as the fine quantizer that resolves 4-bits. By combining a low frequency counter and a delay line TDC, the two-step TDC thus achieves both energy efficiency and a large dynamic range. Detailed implementation of the TDC is discussed in [1].

The output of the ADC is a sequence of time duration measurements s_i, which represent the relative position of the output pulse in every ramp period. Since the signal information is encoded into the position of the pulse, the ADC is called pulse position modulation ADC. The starting points of the ramps are given by $\tau_i = (i-1)T$, where T is the period of the reference ramp signal. The crossover times are $t_i = \tau_i + s_i$. If we assume the slope of the ramp is a constant m, the signal amplitude at the crossover times is $y_i = ms_i$. In this way, from the output $\{s_i\}$ of the PPM ADC, we can calculate the sample set $\{(t_i, y_i), i = 1, 2, ..\}$. Note that T is also the average sampling period of the ADC, because the ADC takes one sample within every interval of T seconds.

Non-uniform signal dependent sampling: If we make the approximation that y_i are samples at uniform time points τ_i instead of the non-uniform t_i, we see harmonic distortion in the frequency spectrum of the recovered signal. Linear low-pass filtering is a straightforward conventional technique for constructing uniform samples from non-uniformly sampled information. According to [43] an oversampling factor of at least 8 is needed to use the traditional low pass filtering technique.

Another approach is to use a time-varying iterative non-linear reconstruction method, (as described in [1]) which allows the signal to be sampled closer to the Nyquist rate. Let us represent measurement vector y as $y = Sx$ where S is the non-uniform sampling operator. Let operator P represent a low pass filter with cut-off frequency tuned to Nyquist frequency. The algorithm described in [1] is as follows:

$$x_0 = y = Sx$$

$$x_1 = Px_0 = PSx$$

$$x_{i+1} = P(y - Sx_i) + x_i, \text{ for } i = 1, 2, ..$$

It is easy to see that $x_i = PS(\sum_{k=1}^{i}(I - PS)^k)x$, where I is the identity operator and $(I - PS)^0 = I$. $\text{Lim}_{i\to\infty} \sum_{k=1}^{i}(I - PS)^k = (PS)^{-1}$ and thus $\text{Lim}_{i\to\infty}x_i = (PS)(PS)^{-1}x = x$. When the PPM ADC is operated at Nyquist rate, applying low pass filter to non-uniform samples causes harmonic distortion which can be corrected through the iterations. However below Nyquist rate, applying a low pass filter to non-uniform samples causes severe aliasing in the frequency domain which cannot be rectified through iterations. In other words, The operator PS cannot be inverted through the algorithm used. Thus, the method still requires the sampling frequency

to be above the Nyquist rate (the oversampling factor of 8 is brought down to 2). Further, a sufficient condition of $s_i < T/4$ is required to obtain a stable sampling set [44]. If this condition is relaxed, there is no guarantee that the algorithm converges. For sampling rates below the Nyquist rate, the method diverges. Our goal is to convert the PPM ADC into a compressive sampling ADC so that the signal can be recovered from samples acquired at sub-Nyquist rate. The sampling system and the reconstruction algorithm are co-designed to achieve this.

A Regular PPM ADC at sub-Nyquist sampling rate: A straight-forward way to operate a PPM ADC as a compressive sampling ADC is to increase its average sampling period T (which is also the reference ramp period). The sampling frequency F can be brought down to a value $F < F_N$, where F_N is the Nyquist frequency of the input signal. We refer to this sampling architecture as the *regular* PPM ADC. We use the algorithm proposed in Section 2.7 for reconstruction. However, since the time points t_i (calculated in Section 2.3.1) are non-random and highly signal dependent, the resultant measurement matrix B is also non-random and thus disobeys the design rules of random sampling algorithms. In order to fit into the compressive sensing framework and to meet the criteria for successful signal reconstruction, we need to make some modifications to the PPM ADC sampling system. A random sampling scheme is introduced in the next section.

2.3.2 The *random* PPM ADC Design

Simple theoretical results from sparse approximation state that a low correlation between different columns of a measurement matrix indicates the possibility of better signal recovery [6]. Albeit crude, this is one of the elementary methods for

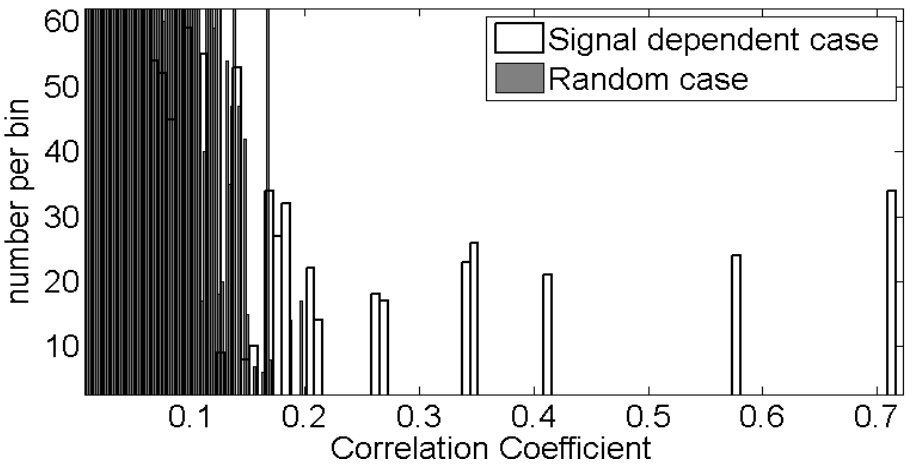

Figure 2.3: Histogram of Correlation Coefficients between different pairs of columns of a signal dependent measurement matrix and a random measurement matrix (of size 15 x 40). The y-axis represents the number of correlation coefficients that fall in any particular bin of coefficient values.

evaluating a measurement matrix with respect to its reconstruction properties. Consider the following simple experiment to motivate the introduction of randomness into the PPM ADC. Let B^c be the measurement matrix that relates the DFT of the input signal to the samples obtained by the PPM ADC at sub-Nyquist rate with an average sampling period of T. As discussed, the time points t_i at which a signal is sampled by the PPM ADC are signal dependent. Now let B^r be the corresponding measurement matrix, when in each interval $[(i-1)T, iT]$ of length T, the signal is sampled at a time point t_i that is uniformly distributed on $[(i-1)T, iT]$[4]. Ideally, we want any measurement matrix B to be orthonormal ($B^H B = I$) so that the input signal can be easily recovered as $B^H y = B^H BX = X$. An orthonormal matrix is characterized by a zero correlation between any two columns of the matrix. Fig. 2.3 plots the histograms of correlation coefficients between different columns, for both the signal dependent B^c and the random B^r matrices. As can be seen from the figure, in the case of a signal dependent B^c several columns have high correlation

[4]Note that in the actual implementation of the random PPM ADC, we don't have complete control over $t_i = \tau_i + s_i$, and so we randomize τ_i instead.

coefficients. On the other hand for the random matrix B^r, coefficients are all distributed in the left region of the plot. Intuitively, since B^r achieves closer to zero correlation coefficients when compared to B^c, it is more "orthonormal" than B^c and is expected to lead to a better signal recovery.

Motivated by this observation, we introduce randomness into the PPM ADC system. We convert the ramp starting times τ_i (which are deterministic in regular PPM ADC) into random variables. More specifically, let $\tau_i - (i-1)T \sim \text{Uniform}[0, T], \forall i$. That is, in each interval $[(i-1)T, iT]$ of length T, the reference ramp has a random starting point. We call this architecture *random* PPM, as the ramp starting times are now randomly and independently chosen. As before, the crossover times are $t_i = \tau_i + s_i$ and the signal amplitude at the crossover times is $y_i = ms_i$, where m is the slope of the ramp.

Assume that the duration of the reference ramp, that is the time for which the ramp is greater than zero in each interval $[(i-1)T, iT]$, is given by cT for some $0 < c < 1$. For the original PPM, $c \leq 0.25$, so as to satisfy the stability condition for the reconstruction algorithm presented in [44]. Choosing $\tau_i - (i-1)T \sim \text{Uniform}[0, T], \forall i$ can cause overlap between adjacent ramps. For example, when $\tau_1 = T$ and $\tau_2 < T + cT$ there is an overlap. There are two ways to deal with this issue. The first is to adjust the distribution of τ_i as $\tau_i - (i-1)T \sim \text{Uniform}[0, T - cT], \forall i$. The implemented prototype random PPM ADC uses this adjustment. Another way is to employ a second sampling system. The sampling systems each produce a ramp in alternate periods and sample the input alternatively in each period. Thus there will be no overlap in the ramps and the original choice of distribution for τ_i can

be maintained, that is $\tau_i - (i-1)T \sim \text{Uniform}[0,T], \forall i$. Note that τ_i are all still independently chosen. Also note that the net power consumption can be kept almost same as before since the two samplers would sample at half the rate as before. Even though we do not actually use two sampling systems in the prototype random PPM ADC, we assume that the overlap between adjacent ramps is allowed for theoretical simplicity. The presented "mathematical framework" thus closely matches the implementation (actual or thought experiment).

The time points at which the random PPM ADC samples the signal are given by $t_i = \tau_i + s_i$. To further aid the analysis, we assume that the phase ϕ of the input signal $x(t+\phi)$, is uniformly distributed in $[0, 2\pi]$. This induces a probability distribution on s_i. The probability density function (pdf) of t_i can be obtained by convolving the pdfs of τ_i and s_i (since τ_i and s_i are independent for all i). Dropping the i for convenience, let $h(r), 0 \leq r \leq cT$ be the pdf of s (not to be confused with sparsity of the signal s). Recall that cT is the on-time of the reference ramp in each period. The pdf of τ is $p_\tau(r) = 1/T, 0 \leq r \leq T$. Thus convolving the two it is easy to see that pdf of $t = \tau + s$ is given by

$$p_t(q) = \begin{cases} \frac{H(q)}{T}, & 0 \leq q \leq cT \\ \frac{1}{T}, & cT \leq q \leq T \\ \frac{1-H(q-T)}{T}, & T \leq q \leq T + cT \end{cases}$$

where $H(q)$ is the cumulative distributive function of the random variable s. An example of $h(r)$ and $p_t(r)$ is shown in the Fig. 2.4 below. These results are used in the proof of Lemma II.1 in Sec. 2.5.

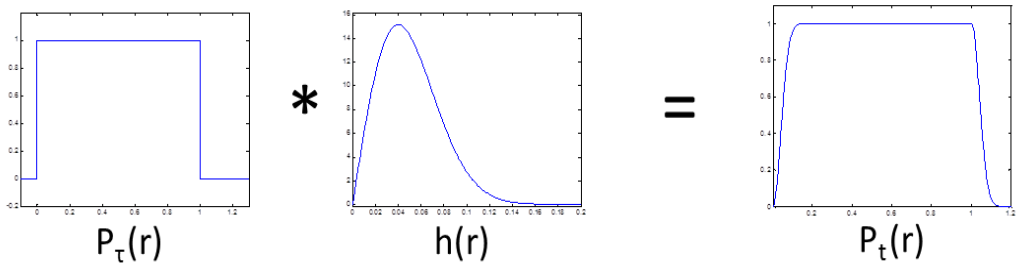

Figure 2.4: Example probability distribution functions (pdfs) of τ, s and $t = \tau + s$

2.4 The *random* PPM ADC Implementation

Fig. 2.5 shows a block diagram of the random PPM ADC. The individual blocks are explained in detail in the following sections. A random clock generator block produces two outputs, a *start* signal (which goes high at each τ_i) and a random clock. The reference ramp is generated only when the *start* signal is high. The comparator compares the output of the ramp generator with the input signal and generates a *stop* signal when the ramp voltage exceeds the input signal. The random clock acts as the time reference in the time-to-digital conversion (TDC) block. The two-step TDC (with a 5-bit coarse quantizer and a 4-bit fine quantizer) measures the time elapsed between the rising edges of the *start* and the *stop* signal. A synchronizer ensures correct alignment of the coarse and the fine time measurements. The ramp generator, comparator and the two-step TDC are implemented in 90 nm digital CMOS, while the random clock generator is implemented on an FPGA.

Some of the key timing signals, shown in Fig. 2.6, provide a comparison of operation between a regular PPM ADC and the random PPM ADC. The regular PPM ADC receives a regular periodic clock with period T_{clk}. The *start* signal of a regular PPM ADC goes high at the beginning of each repetition period T. On the other

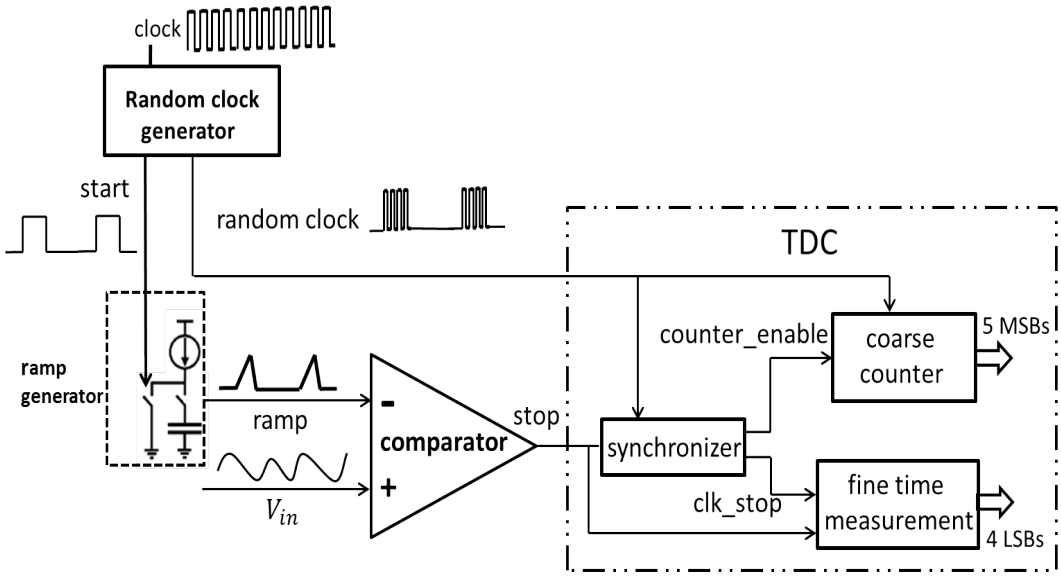

Figure 2.5: The random PPM ADC block diagram along with the TDC building blocks

hand, the random PPM receives a random clock, which consists of regular clock cycles only when the *start* signal is high. The *start* signal for the random PPM ADC goes high after a random time t_{RAND} in each interval (of length T). The *start* signal remains high only for a time t_H and then goes low for the rest of the interval. The time t_H is related to the slope m of the ramp such that the ramp covers the entire voltage range of the input signal in a time t_H. During the time $[t_{RAND}, t_{RAND} + t_H]$ when the *start* is high, the ramp is generated and when it crosses the input signal, the random PPM ADC makes one measurement (denoted as s_1 in the figure). This process repeats in every interval $[(i-1)T, iT]$, $i = 1, 2, ...$ Therefore, the average sampling frequency of the ADC is $F = 1/T$.

2.4.1 Ramp Generator

The ramp generator circuit is shown in Fig. 2.7. Charging a capacitor with a constant current produces the ramp signal. Cascoded PMOS transistors M3 and M4 implement the current source while M1 and M2 are the digital switches that control

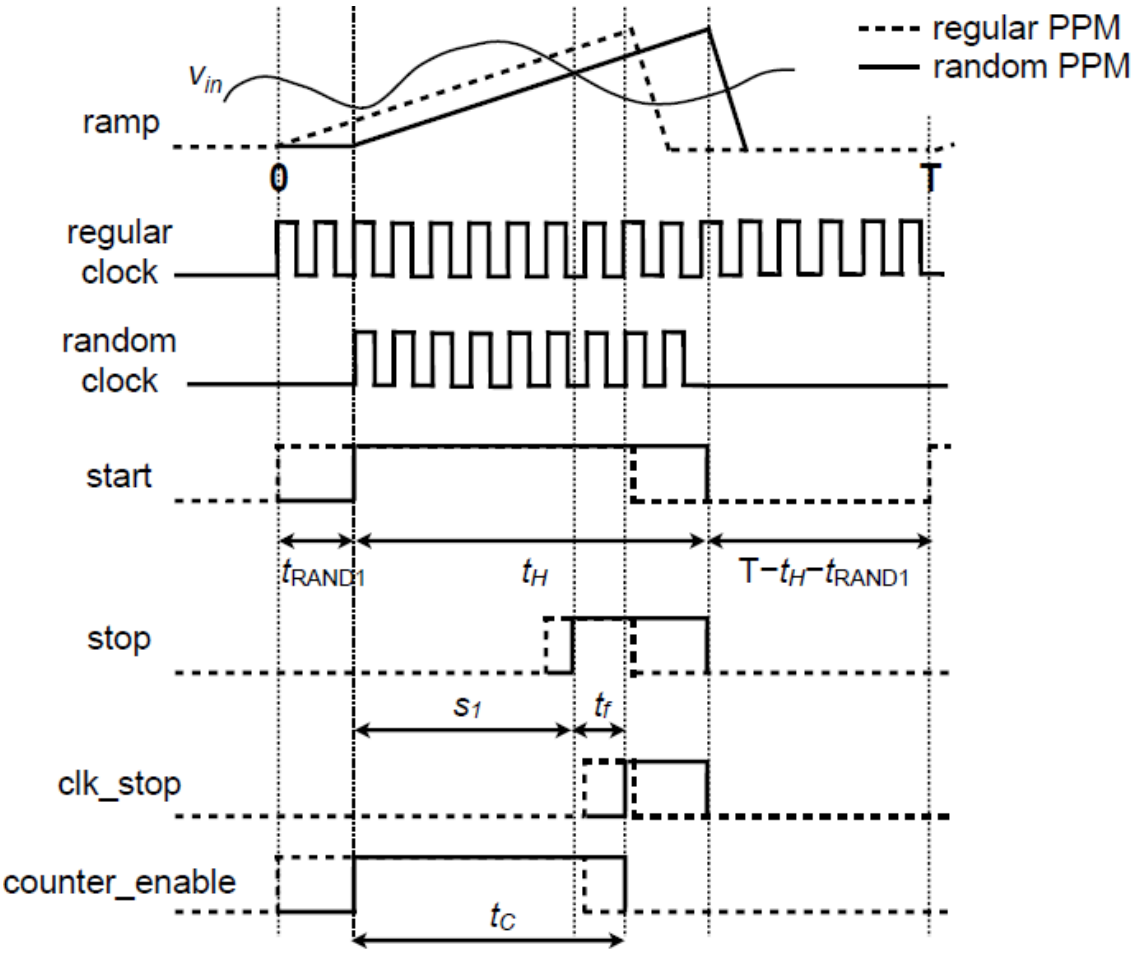

Figure 2.6: Timing signals and comparison of operation between a regular PPM ADC and the random PPM ADC

capacitor charging. The switches are, in turn, controlled by the *start* signal. The capacitor discharge is achieved simply with a switch to ground [1].

2.4.2 Comparator

The comparator is continuous time and is made up of two stages. The circuit is shown in Fig. 2.8. The first stage is a differential to single amplifier with a PMOS input pair. This is followed by an NMOS common source stage. The PMOS input pair operates in the subthreshold region. This is done to minimize power consump-

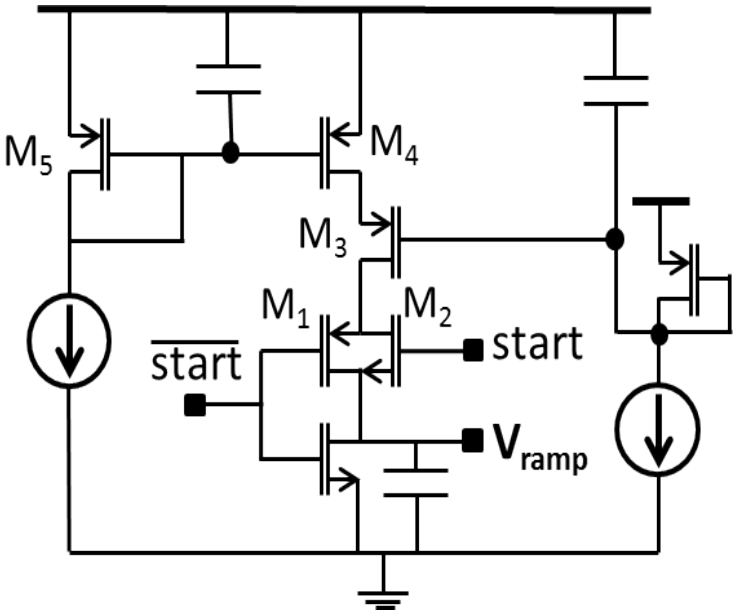

Figure 2.7: The ramp generator which is a component of the ADC in [1]

tion and to provide a larger input common mode range, which allows for a larger dynamic range in the ramp [1].

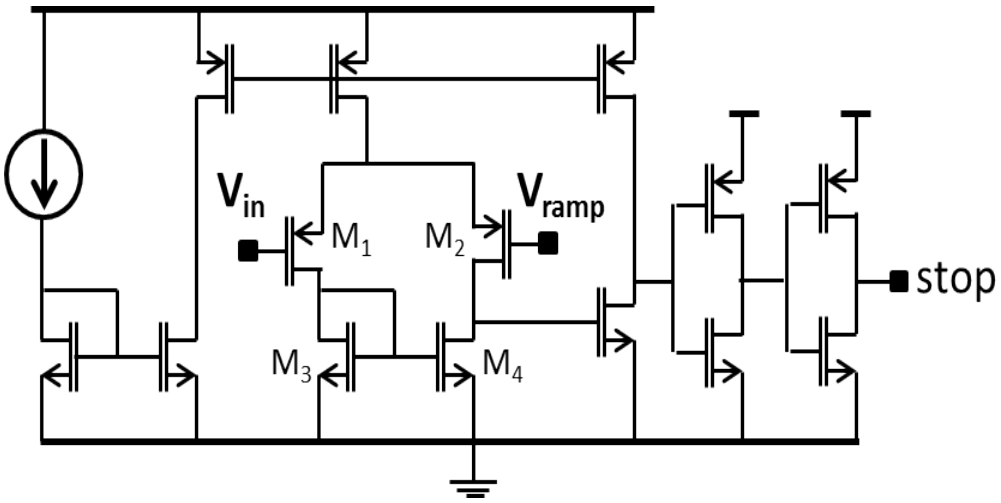

Figure 2.8: Schematic of the comparator, a component of the ADC in [1]

2.4.3 Random clock and start generator

The random waiting times t_{RAND}'s (in Fig. 2.6) are produced using a linear feedback shift register (LFSR) system as shown in Fig. 2.9. Although the output of this system is only pseudo-random, with a large bit length sufficient randomness is achieved. The LFSR bit string is initialized with a non-zero seed. This bit string gives the number of regular clock cycles that the *start* signal initially remains low, i.e., $t_{RAND} = (\text{LFSR})T_{clk}$, where LFSR stands for the (integer) value of the bit string. The *start* signal then goes high and stays high for a time t_H (which is chosen such that mt_H is greater than the input signal voltage range. t_H is also chosen to be a multiple of T_{clk}). To complete the interval of length T, the *start* signal is kept low for an additional $T - t_H - t_{RAND}$ seconds as shown in Fig. 2.6. Once a complete interval of length T has elapsed, the LFSR sequence is advanced to its next state and the same process is repeated with the new value of LFSR (thus, a new $t_{RAND} = (\text{LFSR})T_{clk}$). The random clock is produced by gating the *start* signal with the regular clock as shown in Fig. 2.9. The rising edges of *start* and the random clock are thus synchronized. Note that two short bit length LFSR systems can be coupled to produce a pseudo-random sequence with sufficiently large period.

2.4.4 The Time-to-Digital Converter

The two-step TDC [1] measures the time interval (s_1 in the Fig. 2.6) between the rising edges of the *start* signal (which is synchronous with the random clock), and the *stop* signal generated by the comparator. To enable correct alignment of the coarse and fine time measurements, the synchronizer block generates two additional signals, *clk_stop* and *counter_enable*. The *clk_stop* signal is set by the arrival of the second

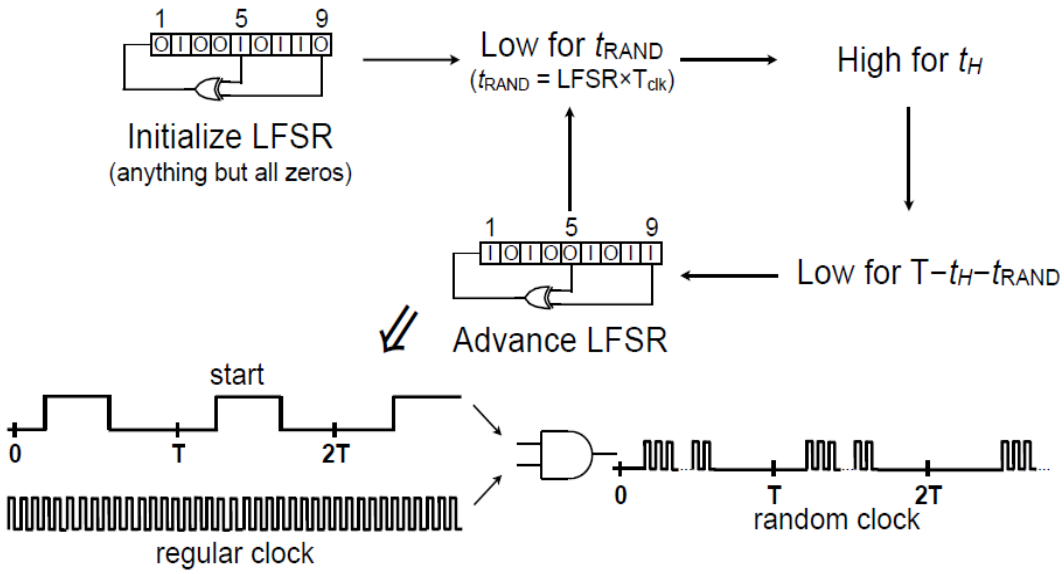

Figure 2.9: Random start signal and random clock generation

rising edge of the clock after the *stop* signal. The *counter_enable* signal is set by *start* and reset by *clk_stop*. A 5-bit counter (which is the coarse quantizer) measures t_c (in Fig. 2.6), which is the number of clock cycles elapsed while the *counter_enable* signal is high. The slope of the ramp is designed such that t_c is always less than 32 clock cycles. The fine TDC measures the time t_f between the *stop* signal and *clk_stop* signal rising edges. The overall TDC output is $s_1 = t_c - t_f$. The fine TDC consists of a 32-element delay line, spanning two full clock cycles (the fine TDC thus divides one clock cycle into 16 equal slices and resolves 4 LSBs).

2.5 The Reconstruction Problem

We now formulate the problem of reconstructing the input signal from samples collected by an ADC (regular PPM or random PPM). The samples are assumed to be collected at a sub-Nyquist rate. Let an N-length vector X represent the input signal in the Fourier domain. Let K $(K < N)$ be the number of measurements taken by the ADC. Let $(t_i, y_i), i = 1, .., K$ denote the measurements obtained from

the output of ADC (see Section 2.3). The time t_i is the i^{th} time point at which the ADC samples the input signal and y_i is the signal amplitude at that time. Note that $\frac{K}{N} = \frac{F}{F_N} < 1$, where $F = 1/T$ is the average sampling frequency of the ADC and F_N is the Nyquist rate of the input signal. We relate the input signal X with the measurement vector y through the equation $BX = y$, where B is the measurement matrix. The goal is to solve for X from $BX = y$. Note that X is the N-point DFT of the time domain input signal x. The reconstruction is done in the frequency domain, as the input signal is assumed to be sparse in frequency domain as indicated in the following input signal model.

2.6 The signal model

In this paper we focus only on a subset of band limited signals that are band limited to $[-W, W]$. The Nyquist rate of the input signal space is $F_N = 2W$. If the input signal is sampled at Nyquist rate for a time of t_{Total}, then the number of samples $N = F_N t_{Total}$. We assume that the input signal is s-sparse or s-compressible in the frequency domain. A signal is called s-sparse in the frequency domain, if the DFT of the signal samples at Nyquist rate has only s non-zero terms. A signal is called s-compressible[5] in frequency domain, if the sorted list of its DFT coefficients has only s significant or dominant terms, compared to which the other terms are negligible. The input signal can be expressed as a linear combination of complex exponentials as follows:

$$x(t) \approxeq \sum_{m=1}^{s} c_m \exp(j2\pi f_m t)$$

where $f_m, m = 1, .., s$ are the s dominant frequencies which lie in the interval $[-W, W]$ and c_m are the corresponding coefficients. We further assume that the input signal is

[5]We call X, s-compressible, if it is well approximated as a s-sparse signal, $||X - X_{(s)}||_2 \leq C.s^{-\alpha}$ for some constants C and $\alpha > 0$, where $X_{(s)}$ is the s-sparse signal that best approximates X.

real, hence s is even and one set of frequencies are the negative of the other set. Some practical signals that are frequency-sparse include frequency-hopping communication signals, narrowband transmissions with an unknown carrier frequency that can lie anywhere in a wide band, communication to submarines, radar [45] and geophysical [46] signals such as slowly varying chirps, etc.

2.6.1 The measurement matrix

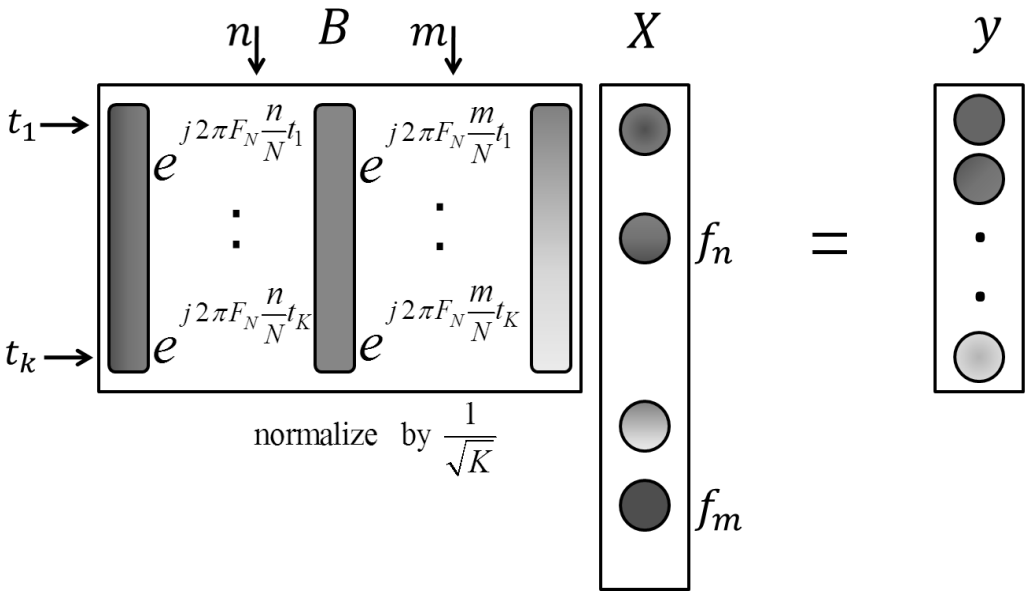

Figure 2.10: Measurement matrix

To determine whether a successful signal recovery is possible from $BX = y$, we analyze the properties of the measurement matrix B, which is shown in Fig. 2.10. The matrix B can be intuitively constructed by making the observation that if f is the only frequency with a non-zero coefficient in the DFT X, i.e., in time domain $x(t) = exp(j2\pi ft)$, then the samples of $x(t)$ at time points t_i are given by $\{exp(j2\pi ft_i), i = 1, .., K\}$. Putting $f = (n/N)F_N$ (as in a N-point IDFT), the samples form the n^{th} column of the measurement matrix, for $n = [-N/2 : N/2 - 1]$ (if N even) or $[(-N - 1)/2 : (N - 1)/2]$ (if N odd). Hence, for a given i and n,

$B_{i,n} = \exp(j2\pi \frac{n}{N} F_N t_i)$. It is to be noted that B is not a sub-matrix of the N-point IDFT matrix, since t_i are non-uniform and do not lie on any Nyquist grid.

We now look at the correlations between different columns of the random PPM measurement matrix B. Let $C_{nm} = B_m^H B_n$ denote the correlation between the n^{th} and m^{th} columns of B. The Lemma II.1 provides a tight upper-bound, on the order of $1/N$, for the magnitude of expected correlation between different columns of B. A small expected correlation implies a better signal recovery, as discussed in Section 2.3.2 and illustrated in Fig. 2.3.

Lemma II.1. *Let* $I_f = [-N/2 : N/2 - 1]$ *(if* N *even) or* $[-(N-1)/2 : (N-1)/2]$ *(if* N *odd). For* $n, m \in I_f$ *and* $n \neq m$,

$$|\mathbb{E}(C_{nm})| \leq \mathrm{O}\left(\frac{1}{N}\right) \tag{2.1}$$

Proof.

$$\begin{aligned}\mathbb{E}(C_{nm}) = \mathbb{E}(B_m^H B_n) &= \mathbb{E}\left(\sum_{i=1}^{K} \frac{1}{K} \exp(j\frac{2\pi}{NT_N}(n-m)t_i)\right) \\ &= \sum_{i=1}^{K} \frac{1}{K} \mathbb{E}\left(\exp(\theta t_i)\right)\end{aligned}$$

where $\theta = \frac{j2\pi(n-m)}{NT_N}$ for convenience and $T_N = 1/F_N$. Now, using the distribution of t_1 derived in Sec. 2.3.2, it can be proven that,

$$\mathbb{E}\left(\exp(\theta t_1)\right) \leq (\exp(\theta T) - 1)\frac{\exp(\theta c T)}{\theta T}$$

Now it is easy to obtain that

$$\mathbb{E}\left(\exp(\theta t_i)\right) = \exp(\theta(i-1)T)\mathbb{E}\left(\exp(\theta t_1)\right)$$

Hence,

$$\begin{aligned}\mathbb{E}(C_{nm}) &\leq \frac{1}{K}\left(\sum_{i=1}^{K}\exp(\theta(i-1)T)\right)\mathbb{E}\left(\exp(\theta t_1)\right)\\ &= \frac{1}{K}\frac{\exp(\theta KT)-1}{\exp(\theta T)-1}\mathbb{E}\left(\exp(\theta t_1)\right)\\ &= \frac{\exp(\theta KT)-1}{\theta KT}\exp(\theta cT)\end{aligned}$$

Now (using $|\exp(\theta cT)| = 1$ and then $NT_N \leq KT < (N+1)T_N$) we have,

$$\begin{aligned}|\mathbb{E}(C_{nm})| &\leq \left|\frac{\exp(\theta KT)-1}{\theta KT}\right|\\ &= \operatorname{sinc}\left((n-m)\frac{KT}{NT_N}\right)\\ &\leq \operatorname{sinc}((n-m)(1+1/N))\\ &\leq 1/(N+1)\\ &\leq 1/N\end{aligned}$$

□

2.7 The Reconstruction Algorithm

The random PPM ADC samples the signal at a rate proportional to its finite rate of innovation, defined as the number of degrees of freedom per unit time [47]. For the signal model considered in this paper, the rate of innovation is given by s, the number of frequencies present in the signal. Algorithms have been proposed in [47] that can recover the s frequencies and their coefficients by using only $2s$ consecutive uniform samples from the signal. However, these algorithms cannot be applied

with the random PPM ADC as they require the samples to be uniformly spaced at Nyquist rate. Also, the measurement matrix B associated with random PPM ADC, is a signal-dependent non-uniform random Fourier matrix, and as such, does not necessarily satisfy the Restricted Isometry Property (RIP) assumed in [7] or the conditions assumed in [8]. This leads to the need to develop different algorithms with different theoretical analysis. A probabilistic approach is presented in Section 2.7.1.

We call the developed reconstruction algorithm, Periodic Random Sampling reconstruction (PRSreco). A pseudo-code for the PRSreco algorithm is presented in Table 2.1. From Lemma II.1, we see that correlations between different columns of B are small on average. Hence, $B^H y$ is a good approximation to the signal X. In particular, the largest components in $B^H y$ provide a good indication of the largest components in X. The algorithm applies this idea iteratively to reconstruct an approximation to the signal X. At each iteration, the current approximation induces a residual, which is the part of the signal that has not been approximated yet. The current approximation vector $\tilde{X}$ is initialized to a zero vector and the residual is initialized to the measurement vector y. For a vector z, supp(z) is defined as set of indices of the non-zero elements of z and $z_{(s)}$ stands for the best s-term approximation[6] of z. For an index set $T \subset \{1, 2, .., N\}$, z_T stands for a sub-vector of z containing only those elements of z that are indexed by T. Similarly B_T stands for a sub-matrix of B containing only the columns of B indexed by T. The algorithm initially obtains an estimate for the dominant frequencies in the signal through least squares and then refines the estimate of the set of dominant frequencies and their coefficients in an iterative fashion.

[6]The best s-term approximation of a vector z can be obtained by equating all the elements of z to zero, except the elements that have the top s magnitudes.

PRSreco algorithm
input: N (signal length), s (sparsity), (t_i, y_i),$i = 1, 2, .., K$.
output: $\tilde{X}$ (s-sparse approximation to X, length N)
$\tilde{X} = \underline{0}$, residual $r^{(0)} = y$ $T =$supp$\left\{[B^H y]_{(2s)}\right\}$ $\tilde{X}_T = \left(B_T^H B_T\right)^{-1} B_T^H y$ (Least Squares) $r^{(0)} = r^{(0)} - B_T \tilde{X}_T$ for $i = 0, 1, 2, ..$ $\tilde{X}^{(i+1)} = \tilde{X} + B^H r^{(i)}$ $\tilde{X} = [\tilde{X}^{(i+1)}]_{(s)}$ $r^{(i+1)} = y - B\tilde{X}$ until $\|\|r^{(i+1)}\|\|_2$ does not vary within a tolerance θ.

Table 2.1: The Periodic Random Sampling Reconstruction (PRSreco) Algorithm

The computationally intensive step of least squares is performed only once in the PRSreco algorithm. The least squares is implemented using the accelerated Richardson iteration [48] with runtime of O($sK\log(2/e_t)$) where e_t is a tolerance parameter. The structure of the measurement matrix lends us to use the inverse NUFFT [49] with cardinal B-spline interpolation for forming the products of the form $B^H r$, in a runtime of O($N\log N$). Hence the total runtime of the algorithm is dominated by O($IN\log N$) where I is the number of iterations.

2.7.1 Analysis of Algorithm

Lemma II.2 says that the estimators of coefficients of X in the PRSreco algorithm produce close to correct values and their second moments (variances) are bounded. The results of Lemma II.2 and Lemma II.1 are used to prove Theorem II.3.

Lemma II.2. *If number of measurements $K = \mathrm{O}(s/\epsilon^2)$ then for any s-sparse (or s-compressible) vector X, each estimate of the form $\tilde{X}_m = B_m^H BX$ for $m = 1, 2, .., N$, satisfies*

$$\mathbb{E}(\tilde{X}_m) = X_m \pm \mathrm{O}\left(\frac{1}{N}\right) ||X||_1 \tag{2.2}$$

$$\mathrm{Var}(\tilde{X}_m) \leq \frac{\epsilon^2}{s} ||X||_2^2 \tag{2.3}$$

Proof. For any vector X,

$$\begin{aligned}\mathbb{E}(\tilde{X_m}) &= \mathbb{E}\left(\sum_{i=1}^{N} B_m^H B_i X_i\right) = \mathbb{E}\left(X_m + \sum_{i=1, i\neq m}^{N} C_{im} X_i\right) \\ &= X_m + \sum_{i\neq m} \mathbb{E}(C_{im}) X_i\end{aligned}$$

The required result is now true from Lemma II.1.

Now we will compute Var $(\tilde{X_m}) = \mathbb{E}(\tilde{X_m}^2) - (\mathbb{E}(\tilde{X_m}))^2$. But first, consider the following:

$$\begin{aligned}C_{im}^H C_{\ell m} &= \frac{1}{K^2} \sum_{n=1}^{K} e^{\left(j\frac{2\pi(m-i)t_n}{NT_N}\right)} \sum_{q=1}^{K} e^{\left(j\frac{2\pi(\ell-m)t_q}{NT_N}\right)} \\ &= \frac{1}{K^2} \sum_{n=1}^{K} e^{\left(j\frac{2\pi}{NT_N}(m-i+\ell-m)t_n\right)} \\ &\quad + \frac{1}{K^2} \sum_{n} \sum_{q\neq n} e^{\left(j\frac{2\pi(m-i)t_n}{NT_N}\right)} e^{\left(j\frac{2\pi(\ell-m)t_q}{NT_N}\right)}\end{aligned}$$

After applying expectation, and using that t_n and t_q are independent for $n \neq q$ and also using the Lemma II.1,we see that the second term above can be ignored as it is $O(\frac{1}{N^2})$.

Hence $\mathbb{E}(C_{im}^H C_{\ell m}) \cong \mathbb{E}(C_{\ell i})/K$. We will use this in the expansion for $\mathbb{E}(\tilde{X_m}^2)$ as follows,

$$\begin{aligned}
\mathbb{E}(\tilde{X_m}^H \tilde{X_m}) &= X_m^2 + \sum_{\ell=1,\neq m}^{N} \mathbb{E}(C_{\ell m}) X_m^H X_\ell \\
&\quad + \sum_{i=1,\neq m}^{N} \mathbb{E}(C_{mi}) X_i^H X_m + \sum_{i\neq m} \sum_{\ell \neq m} \frac{\mathbb{E}(C_{\ell i})}{K} X_i^H X_\ell \\
&\leq X_m^2 + \sum_{\ell=1,\ell\neq m}^{N} \mathbb{E}(C_{\ell m}) X_m^H X_\ell \\
&\quad + \sum_{i=1,i\neq m}^{N} \mathbb{E}(C_{mi}) X_i^H X_m + \sum_{i=1,i\neq m}^{N} \frac{1}{K} X_i^2
\end{aligned}$$

since, by Lemma II.1,$\mathbb{E}(C_{\ell m}) \leq 1/N$, is negligible for $\ell \neq m$. We can similarly obtain the expansion $(\mathbb{E}(\tilde{X_m}))^2 = X_m^2 + \sum_{\ell=1,\ell\neq m}^{N} \mathbb{E}(C_{\ell m}) X_m^H X_\ell + \sum_{i=1,i\neq m}^{N} \mathbb{E}(C_{mi}) X_i^H X_m + \sum_{i=1,i\neq m}^{N} \sum_{\ell=1,\ell\neq m}^{N} \mathbb{E}(C_{mi})\mathbb{E}(C_{\ell m}) X_i^H X_\ell$. The last term can be ignored (assuming signal sparsity $s \ll N$). Now,

$$\begin{aligned}
\mathrm{Var}(\tilde{X_m}) &= \mathbb{E}(\tilde{X_m}^H \tilde{X_m}) - (\mathbb{E}(\tilde{X_m}))^2 \\
&\leq \sum_{i=1,i\neq m}^{N} \frac{1}{K} X_i^2 \leq \frac{\epsilon^2}{s} ||X||_2^2
\end{aligned}$$

□

Once the PRSreco algorithm gets an approximation $\tilde{X}$ of X, it subtracts the contribution of the current approximation from the measurements and proceeds to recover the leftover signal $X - \tilde{X}$. As we move on to higher iterations of the algorithm, the energy in the leftover signal goes down, bringing down the upper-bound on the variance of the estimators (from Lemma II.2 applied to $X - \tilde{X}$). Thus a better approximation is obtained for the signal X in each higher iteration until the

required tolerance is reached or the algorithm converges. Please refer to the proof of Theorem II.3 for further details. Theorem II.3 offers an error guarantee for a signal recovered using the PRSreco algorithm and establishes the conditions on the sub-sampling ratio K/N that can be achieved using the random PPM ADC. If X is s-sparse and there is no noise in the measurements obtained from the random PPM ADC (operating at a sub-sampling ratio of K/N), then from Theorem II.3, signal X can be recovered exactly. If the measurements are corrupted by some noise (e.g. quantization noise), the ℓ_2-norm of the reconstruction error is bounded above by the ℓ_2-norm of the noise.

Theorem II.3. *Let $y = BX + \xi$ be the time domain samples of signal X obtained by the random PPM ADC, where ξ is an arbitrary noise contamination in the measurements and B is the resultant measurement matrix of size KxN. Let the phase[7] ϕ of the time domain input signal $x(t+\phi)$ be uniformly distributed in $[0, 2\pi]$. Suppose $|X_{[s]}|^2 \geq 2\alpha||X||_2^2/s + |X_{[s+1]}|$ for some constant α and a given sparsity parameter s, where $|X_{[i]}|$ is the magnitude of the i^{th} largest element of X. Given the error tolerance in reconstruction θ and $K = \mathrm{O}(s\ \log N/\epsilon^2)$, with probability $> 1-\mathrm{O}(\epsilon^2)$ the algorithm produces an s-term estimate $\tilde{X}$ of signal with the following property,*

$$||X - \tilde{X}||_2^2 \leq \max\left\{\theta^2, \frac{||X - X_{(s)}||_2^2}{1-\alpha} + \frac{c(B)||\xi||_2^2}{1-\alpha}\right\} \tag{2.4}$$

where $X_{(s)}$ is the best s-term approximation of X. The runtime of the algorithm is $\mathrm{O}(IN\log N)$ where I = Number of iterations, with a gross upper bound of $I < \max(\log N, \log(||X||_2/\theta))$. The net storage requirement is $\mathrm{O}(N)+\mathrm{O}(sK)$. The constant $c(B)$ depends on the measurement matrix B.

[7] That is, the time $t = 0$ at which we start to observe the signal, is assumed to be random. This induces a probability distribution on the signal dependent s_i.

Proof. First we will show that the PRSreco algorithm succeeds in identifying the top s terms of the signal. We will then derive the error guarantee.

Let us begin with signal X exactly s-sparse. For simplicity lets assume that $X_i, i = 1, .., s$ are the non-zeros. There exists a $\beta < 1$ such that $|X_i|^2 \geq \beta||X||^2/s$. Let $0 < 2\alpha \leq \beta$. For $i = 1, .., s$, using the Chebyshev inequality we have, $\mathbb{P}\text{r}(|\tilde{X}_i^{(1)} - X_i|^2) \geq \frac{\alpha||X||^2}{s}) \leq \text{Var}(\tilde{X}_i^{(1)})/\frac{\alpha||X||^2}{s} \leq \frac{\epsilon^2||X||^2}{s}/\frac{\alpha||X||^2}{s} = \frac{\epsilon^2}{\alpha}$ (using Equation (2.3) from Lemma (II.2)). Hence

$$\mathbb{P}\text{r}\left(\tilde{X}_i^{(1)}\text{good}\right) = \mathbb{P}\text{r}\left(|\tilde{X}_i^{(1)} - X_i|^2 \leq \frac{\alpha||X||^2}{s}\right) \geq 1 - \frac{\epsilon^2}{\alpha}$$

Let $|X_{min}|$ be the smallest non-zero in X. Again using Chebyshev inequality, for each of $X_i, i > s$ we have $\mathbb{P}\text{r}(|\tilde{X}_i^{(1)}|^2 \leq |X_{min}| - \frac{\alpha||X||^2}{s}) \geq \mathbb{P}\text{r}(|\tilde{X}_i^{(1)}|^2 \leq (\beta - \alpha)\frac{||X||^2}{s}) \geq 1 - \frac{\epsilon^2}{\beta-\alpha} = 1 - \frac{\epsilon^2}{\alpha}($ since $2\alpha \leq \beta)$. Now, define Bernoulli random variables z_i as indicators of failure of the i^{th} coefficient estimator. That is

$$\mathbb{P}\text{r}(z_i = 0) = 1 - \frac{\epsilon^2}{\alpha} = 1 - \mathbb{P}\text{r}(z_i = 1)$$

for all $i = 1, .., N$. Let $Z_1 = \sum_{i=1}^{s} z_i$ and $Z_2 = \sum_{i=s+1}^{N} z_i$. We have

$$\mathbb{P}\text{r}\left(Z_1 > \frac{s}{4}\right) \leq \frac{\mathbb{E}(Z_1)}{s/4} \leq \frac{s\epsilon^2/\alpha}{s/4} = \frac{4\epsilon^2}{\alpha}.$$

Hence $\mathbb{P}\text{r}(\text{No. of good estimators among } \tilde{X_1}^{(1)}, \tilde{X_2}^{(1)}, ..., \tilde{X_s}^{(1)} \geq \frac{3s}{4}) \geq 1 - \frac{4\epsilon^2}{\alpha}$. Note that the factor $1/4$ is chosen as an example to simplify the presentation of the proof. Now lets move on to the 2^{nd} iteration of the algorithm. More than $3s/4$ estimators which were good in the first iteration are still good in the second iteration. This is because the estimator $\tilde{X}_i^{(2)}$ depends on the same random correlations (between B_i and other columns of B) as the estimator $\tilde{X}_i^{(1)}$ from the first iteration. Put the current approximation $\tilde{X} = [\tilde{X}^{(1)}]_{(s)}$ as defined in the PRSreco algorithm (see

Table 2.1). Now for those coefficients whose estimators were not good in the first iteration we have,

$$\mathbb{P}\mathrm{r}\left(|\tilde{X}_i^{(2)} - X_i|^2 \geq \frac{\alpha||X-\tilde{X}||^2}{s}\right) \leq \frac{\epsilon^2||X-\tilde{X}||^2/s}{\alpha||X-\tilde{X}||^2/s} = \frac{\epsilon^2}{\alpha}$$

like before, using the Equation (2.3) from Lemma (II.2) applied to $X - \tilde{X}$. Now define a new $Z_1^{(2)}$ for these estimators. Note that $\mathbb{E}(Z_1^{(2)}) \leq \frac{s\epsilon^2}{4\alpha}$ (since there are less than $s/4$ terms in the definition of $Z_1^{(2)}$). Now as before we have

$$\mathbb{P}\mathrm{r}\left(Z_1^{(2)} > \frac{s}{4^2}\right) \leq \frac{4\epsilon^2}{\alpha}.$$

Hence by the end of second iteration number of good estimators among the $i = 1, .., s$ is $\geq \frac{3s}{4} + \frac{3}{4}\frac{s}{4}$ with a net probability $\geq (1 - \frac{4\epsilon^2}{\alpha})^2$. Going on this way at k^{th} iteration, number of good estimators $\geq (1 - (\frac{1}{4})^k)s$, with probability $\geq (1 - \frac{4\epsilon^2}{\alpha})^k$. Similar statements can be obtained about Z_2, i.e., about the estimators with $i > s$. Hence after sufficient number of iterations (say I), all the estimators are good which implies that all the non-zero terms will be identified by the algorithm with

$$\mathbb{P}\mathrm{r}(\text{Success}) \geq (1 - \frac{4\epsilon^2}{\alpha})^{2I} \approx (1 - \frac{8I\epsilon^2}{\alpha}) = 1 - O(\epsilon^2)$$

after absorbing some constants along with number of iterations I into the number of measurements. If I is the sufficient number of iterations at which all estimators are good, then $(1/4)^I N < 1 \Rightarrow I = 0.5 \log N = \mathrm{o}(\log N)$. Hence an increase in number of measurements by a factor of $\log N$ is required. Note that the above is a gross lower bound for the success probability. In reality since all the estimators are highly dependent, the probability that they will be good together is higher than the product of the individual success probabilities, which is the gross lower bound produced by the above theory.

Now let the signal X be s-compressible (hence not exactly s-sparse). We start with $K = O(s/\epsilon^2)$ as before. Again for simplicity let the first s elements of X be the top s terms. For $i > s$ we assume that $|X_i|^2 \leq \gamma ||X||^2/s$ for some $\gamma < 1$. Let X^h_{min} be the smallest coefficient in the head ($i = 1, .., s$) of X. Similarly let X^t_{max} be the largest coefficient in the tail ($i > s$) of X. All the above arguments hold again except that for $i > s$ the probabilities will involve γ in the following manner. For example in the first iteration,

$$\begin{aligned}
&\mathbb{P}\mathrm{r}\Big(|\tilde{X}_i - X_i|^2 \leq |X^h_{min}| - \tfrac{(\alpha)||X||^2}{s} - |X^t_{max}|\Big) \\
&\quad \geq \mathbb{P}\mathrm{r}\Big(|\tilde{X}_i - X_i|^2 \leq (\tfrac{1}{\beta} - \alpha - \gamma)\tfrac{||X||^2}{s}\Big) \\
&\quad \geq 1 - \tfrac{\beta\epsilon^2}{1-(\alpha+\gamma)\beta} = 1 - \tfrac{\epsilon^2}{\alpha}
\end{aligned}$$

(assuming $0 < 2\alpha \leq \beta - \gamma$). Repeating the arguments from above we show that the algorithm succeeds in identifying the top s-terms.

Now let us prove the error guarantee. Let us assume that $\xi = 0$ for the moment. Lets say the algorithm correctly identifies the position of top s terms in I iterations. For any $k > I$,at iteration $k+1$, $|(\tilde{X}^{(k+1)}_i - X_i)|^2 < \alpha ||X - \tilde{X}^{(k)}||^2/s$ for $i = 1, .., s$. Summing up the s inequalities we get,

$$||\tilde{X}^{(k+1)} - X_{(s)}||^2 \leq \alpha ||X - \tilde{X}^{(k)}||^2$$

where $X_{(s)}$ is the best s-term approximation to X. Now, $||X - \tilde{X}^{(k+1)}||^2 \leq ||X - X_{(s)}||^2 + ||X_{(s)} - \tilde{X}^{(k+1)}||^2 \leq ||X - X_{(s)}||^2 + \alpha ||X - \tilde{X}^{(k)}||^2$. This implies, $||X - \tilde{X}^{(k+1)}||^2 \leq \frac{1-\alpha^{k-I}}{1-\alpha}||X - X_{(s)}||^2 + \alpha^{k-I}||X - \tilde{X}^{(k-I)}||^2$. For k large enough we have,

$$||X - \tilde{X}||^2 \leq \frac{1}{1-\alpha}||X - X_{(s)}||^2$$

This is consistent with Equation 2.4.

Now let $\xi = Bn$ for some vector n. we have $y = B(X + n)$. Following the arguments as before, we have, $||X + n - \tilde{X}||^2 \leq \frac{||X+n-(X+n)_{(s)}||^2}{1-\alpha} \leq \frac{||X+n-X_{(s)}||^2}{1-\alpha}$ (since $(X + n)_{(s)}$ is the best s-term approximation to $X + n$). Now, $||X - \tilde{X}||^2 \leq ||X + n - \tilde{X}||^2 + ||n||^2 \leq \frac{||X+n-X_{(s)}||^2}{1-\alpha} + ||n||^2 \leq \frac{||X-X_{(s)}||^2}{1-\alpha} + \frac{(2-\alpha)||n||^2}{1-\alpha}$. We will have Equation 2.4 by putting $||n||_2 \leq c||\xi||_2$. This is true for some $c(B) < \frac{1}{\sigma_{min}(B)}$ where the denominator is the smallest singular value of B.

□

2.8 Algorithm 2: Median of estimators (MOE)

Note that in the PRSreco algorithm (Sec. 2.7), the input signal was sampled for a total time of $t = N/F_N$ to get K samples. Instead if we sample the signal for a duration of mt, we get m copies of K measurements,with each set of measurements from a block of time t. Assume that the set of top s frequencies in the signal remains the same in all the m blocks of time (their coefficients can change). Then we can take a median over the estimators from different blocks to improve the identification of the top s frequencies. The idea of taking a median instead of mean was used in the count sketch algorithm [50] which estimates the most frequent items in a data stream.

We propose to use the algorithm in Table 2.2 to identify and estimate the top s frequencies in the signal. Let $B(i)$ be the measurement matrix formed (as shown in Section 2.6.1) from the time points in the i^{th} block of time, for $i = 1, .., m$. Similarly let $y(i)$ be the vector of measurements obtained from the i^{th} block.

MOE algorithm
input: N (Block length), m (No. of Blocks), s (sparsity) (t_ℓ, y_ℓ),$\ell = 1, 2, .., mK$.
output: $X(i)$ for $i = 1, .., m$ (signal in each block)
Identification: For $j = 1, .., N$, $\hat{X}_j = \text{median}\big\{\lvert B(1)_j^H y(1)\rvert, .., \lvert B(m)_j^H y(m)\rvert\big\}$ $T = \text{supp}([\hat{X}]_s)$ **Estimation:** For $i = 1, .., m$, $[\tilde{X}(i)]_T = \big(B(i)_T^H B(i)_T\big)^{-1} B(i)_T^H y(i)$

Table 2.2: Algorithm 2 : The Median of Estimators (MOE)

Theorem II.4. *For $m = \text{O}(\ln(\frac{N}{\delta}))$, the MOE algorithm correctly identifies the set T of top s frequencies in the signal with $\mathbb{P}\text{r}(\text{Success}) \geq 1 - \delta$.*

Proof. Note that the arguments in proof of Theorem II.3 hold for all the m blocks of time. Let $\tilde{X}_{ij} = B(i)_j^H y(i)$ for $i = 1, .., m$ and $j = 1, .., N$. Note that $\hat{X}_j$ =median$(|\tilde{X}_{ij}|, i = 1, .., m)$ for $j = 1, .., N$. From Theorem II.3's proof, $\mathbb{P}\text{r}(|\tilde{X}_{ij}| \text{ good}) \geq 1 - \frac{\epsilon^2}{\alpha} = p(\text{say})$, for $i = 1, .., m$. Assuming $p > 0.5$, from Chernoff bound we have $\mathbb{P}\text{r}(\hat{X}_j \text{ good}) \geq 1 - e^{-2m(p-0.5)^2} \geq 1 - \delta'$ for m =O$(\ln(\frac{1}{\delta'}))$. Hence $\mathbb{P}\text{r}(\text{Success}) = \mathbb{P}\text{r}(\hat{X}_j \text{ good for } j = 1, .., N) \geq (1 - \delta')^N \approx 1 - \delta$ for $\delta' = \delta/N$. □

Note that the computationally intensive step of least squares is performed only once (per block of the signal) in both the algorithms. The least squares was implemented using the accelerated Richardson iteration [48] with runtime of O$(sK\log(2/e_t))$ where e_t is a tolerance parameter. The structure of the measurement matrix lends us to use the inverse NUFFT [49] with cardinal B-spline interpolation for forming the products of the form $B^H r$, in a runtime of O$(N\log N)$. Hence the total runtime of PRSreco algorithm is dominated by O$(IN\log N)$ where I is the number of iterations. The per block runtime of the MOE algorithm which has only one iteration

is O(sKlog$(2/e_t)$)+O(Nlog$N$), which is much less than that of the PRSreco algorithm. However as mentioned in section 2.8 to apply the MOE algorithm the signal has to satisfy the required additional condition of maintaining the same dominant set of frequencies throughout the observed time. Also the MOE algorithm processes the signal in blocks of m unlike the PRSreco algorithm. The sampling percentage ($K/N = mK/mN$) is the same for both the algorithms. (100% sampling implies sampling at Nyquist rate.) These statements are summarized in Table 2.3.

	PRSreco	**MOE**
Signal model	Any sparse signal	Same set of dominant s frequencies in all m blocks (coefficients can vary)
Output	Estimate of signal of length N	Estimate of signal of length N on each of m blocks
Total number of operations	O(INlog$N$) per block	O(mNlog$N$) per m blocks
% Sampling	K/N	$mK/mN = K/N$
Operation efficiency	$\frac{\text{O}(IN\text{log}N)}{N}$	$\frac{O(mN\,logN)}{mN}$

Table 2.3: Comparison of the PPMreco and MOE algorithms, used for signal reconstruction with random PPM ADC.

2.9 Experimental Results and Discussion

The regular PPM and the random PPM sampling architectures (described in Section 2.3) are implemented in hardware. The ADCs combined with the reconstruction algorithms are also simulated in MATLAB. A series of experiments compares the performance of the algorithms for both the sampling architectures. The Signal-to-Noise Ratio[8] (SNR), which is defined as the ratio between the signal energy and the reconstruction error, is used as the performance metric to evaluate the quality of the reconstructed signal. MATLAB simulation results are presented

[8]SNR(dB) = 20 log$(||X||_2/||X - \tilde{X}||_2)$, where X is the input signal and $\tilde{X}$ is the output of the algorithm

first and are followed by the experimental results from the hardware implementation.

2.9.1 Simulation results

The finite time resolution t_r of the TDC block in the ADC induces some quantization into the measurements. For the simulation experiments to follow, the quantization is kept at 7 bits (= $\log_2$(ramp duration/t_r), with a ramp duration of 0.25 μsec and $t_r = 2$ nsec). This corresponds to a signal to quantization noise ratio of about 44 dB for an input sinusoid.

Multitone signals In the first experiment we reconstruct multi-tone input signals, which are a linear combination of sinusoids. Each sinusoid has a random phase, comparable amplitude and its frequency is chosen randomly from the Nyquist grid. The Nyquist frequency is 3 MHz whereas the sampling frequency of the ADC is chosen to be 1 MHz, giving a sub-sampling ratio of 0.33. That is, $K/N = 0.33$, where $K(= 150)$ is the number of measurements from the ADC and $N(= 450)$ is the length of input signal, X. The input signal is corrupted by additive white Gaussian noise with varying power, sampled by the two sampling schemes and reconstructed using the PRSreco algorithm. The performance of the algorithms is evaluated by measuring the output SNR. The experiment uses the s-term Nyquist approximation as the benchmark performance, which is defined as the SNR obtained when the signal is sampled at Nyquist rate, quantized at the same quantization level as the ADC and then truncated, in frequency domain, to keep only the s dominant terms. The s-term Nyquist benchmark thus represents the best s-term approximation to the signal in frequency domain. Fig. 2.11(a) plots the mean (of 200 trials) reconstruction output SNRs for signals with 9 tones (corresponding $s/N = 18/450$ and $s/K = 18/150$) and

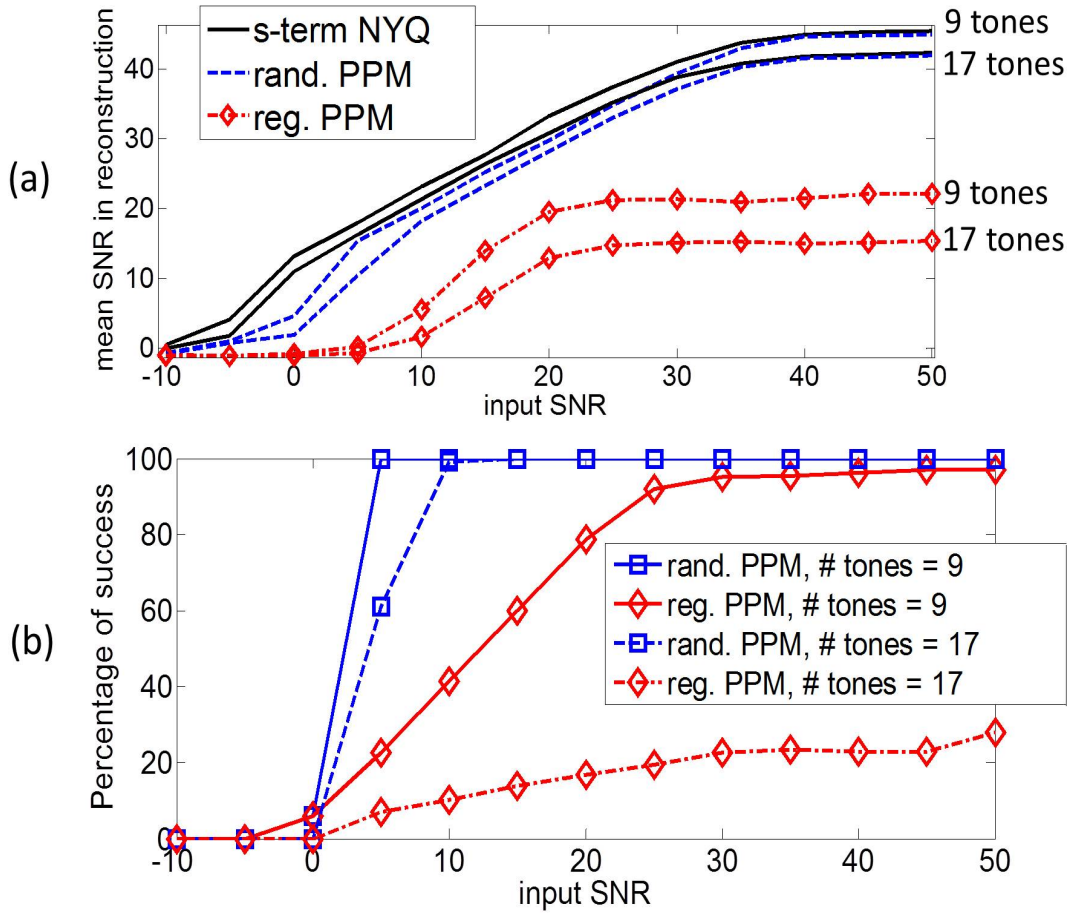

Figure 2.11: (a) Mean output SNR versus input SNR and (b) success percentage (fraction of trails that succeed) versus input SNR for 9-tone and 17-tone signals. The s-term NYQ (Nyquist) benchmark represents the best s-term approximation to the signal in frequency domain. Success means the correct identification of the frequencies of all tones.

17 tones ($s/N = 34/450$ and $s/K = 34/150$).

The experiment demonstrates the better performance of random PPM in two ways. First, random PPM achieves a higher output SNR compared to the regular PPM and is closer to the benchmark[9] performance, owing to the better correlation properties of the measurement matrix (Lemma II.1). The random PPM performance approaches the benchmark as the input SNR increases. Secondly, as the number of tones increases (making the signal less sparse), the random PPM output SNR is un-

[9]The benchmark considers the error in the amplitude of the s tones due to quantization and input noise

affected relative to the benchmark while the output of constant PPM degrades. This indicates that the random PPM design can handle less sparse signals much better than the regular PPM scheme for same number of measurements.

The output SNR can be higher than the input SNR, as the algorithm (like any other greedy pursuit algorithm) only calculates the coefficients of the top s frequencies in the signal and thus inherently filters out the noise at other frequencies. This "denoising" effect decreases as the value of s increases. This explains the degradation in output SNR (of even the benchmark) when the number of tones is increased. After input SNR is high enough, we see a saturation in the output SNR. This can be attributed to the quantization noise in the measurements (which also gets "denoised" to some extent).

Fig. 2.11(b) plots the percentage of trials that achieve success in signal recovery. We call the reconstruction a success when the frequencies of all the tones in the input signal are correctly identified. Once again we observe that random PPM performs much better than the regular PPM. The plot also conforms that mean output SNR is a good indicator of the quality of reconstruction, as it also captures (to some extent) the information about the percentage of success.

Sampling percentage The next experiment reconstructs a single tone signal (randomly chosen frequency, $s/N = 2/450$) with varying number of measurements and noise levels using the PRSreco algorithm. The sub-sampling ratio is defined as the ratio between the sampling rate of the ADC and the Nyquist rate of the signal (which is twice the randomly chosen tone frequency), and can be computed as K/N. The

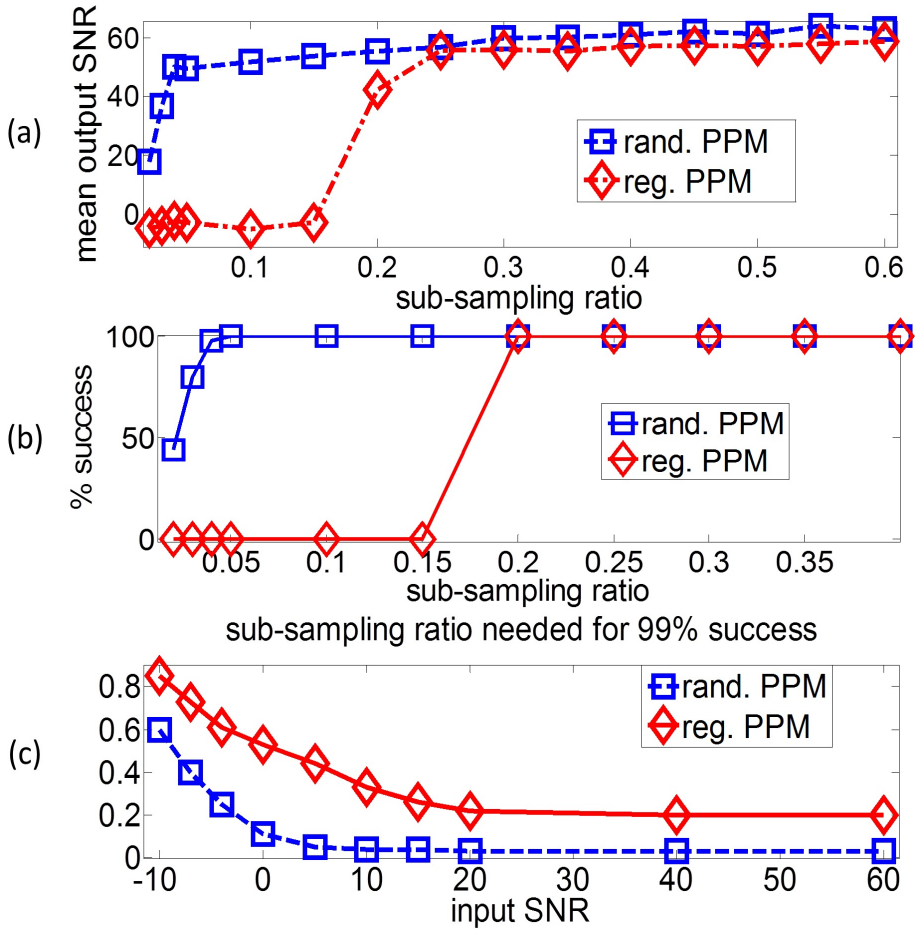

Figure 2.12: Reconstruction of a single tone signal with varying number of measurements (a) with no noise (b) success percentage when no noise (c) sampling needed for 99% success, with noise

sub-sampling ratio needed for at least 99% success (i.e. at least 99% of the total trials succeed in identifying the input signal frequencies correctly) is empirically determined for each input SNR level and is plotted in Fig. 2.12(c). We observe that at all SNR levels the random PPM ADC succeeds with far fewer measurements than the regular PPM. Further, when the input SNR is high enough the sub-sampling ratio needed for success in the random PPM quickly falls to about 3%. This can also be seen in the no-noise (i.e. only quantization noise) case (Fig. 2.12(a),(b)), where the regular PPM scheme breaks down when the sampling rate goes below 20% of Nyquist rate, whereas, the random scheme performs well enough for sampling rates

as low as 3% of the Nyquist rate, indicating much better incoherence properties of the measurement matrix.

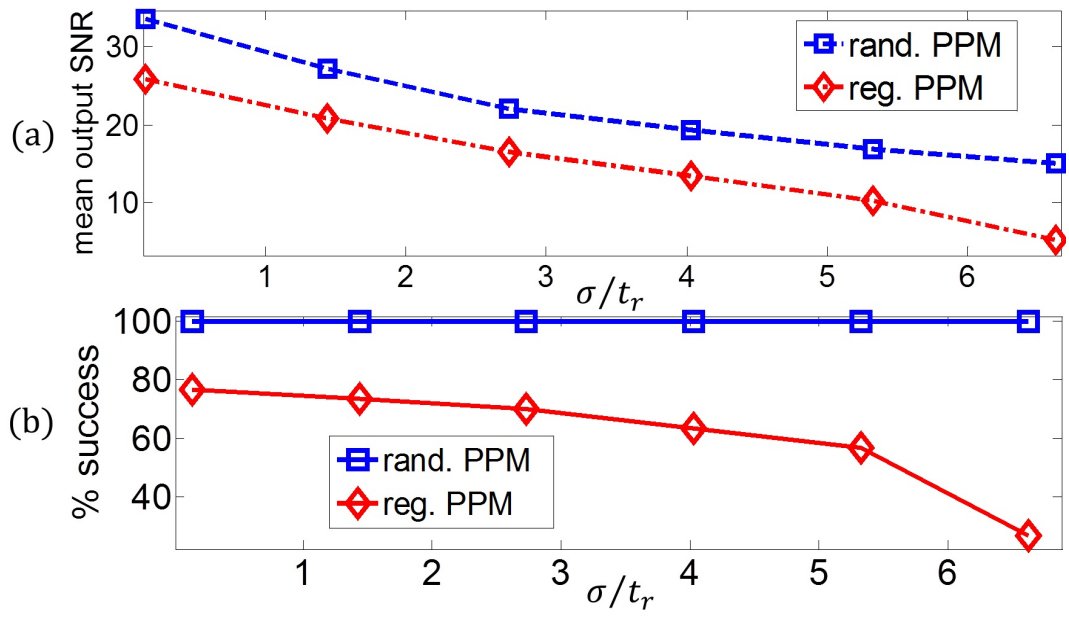

Figure 2.13: Reconstruction of a 11 tone signal with varying amount of time jitter noise

Time jitter noise Noise generated in the ramp and comparator circuits degrades the accuracy of the time measurement. In this experiment, we model the time measurement error as a normal random variable with standard deviation σ. Fig. 2.13 plots the reconstruction results for a random 11-tone signal sampled by both the random and regular PPM ADCs, for varying σ (expressed as a multiple of the finite time resolution t_r of the TDC block). While there is a degradation in the SNR performance of both the ADCs as σ increases, we see that the success percentage for the regular PPM ADC is much more sensitive to time jitter.

Resolution versus sampling rate In this experiment we fix the bit-rate of random PPM ADC, that is the product of ADC quantization (resolution) and its

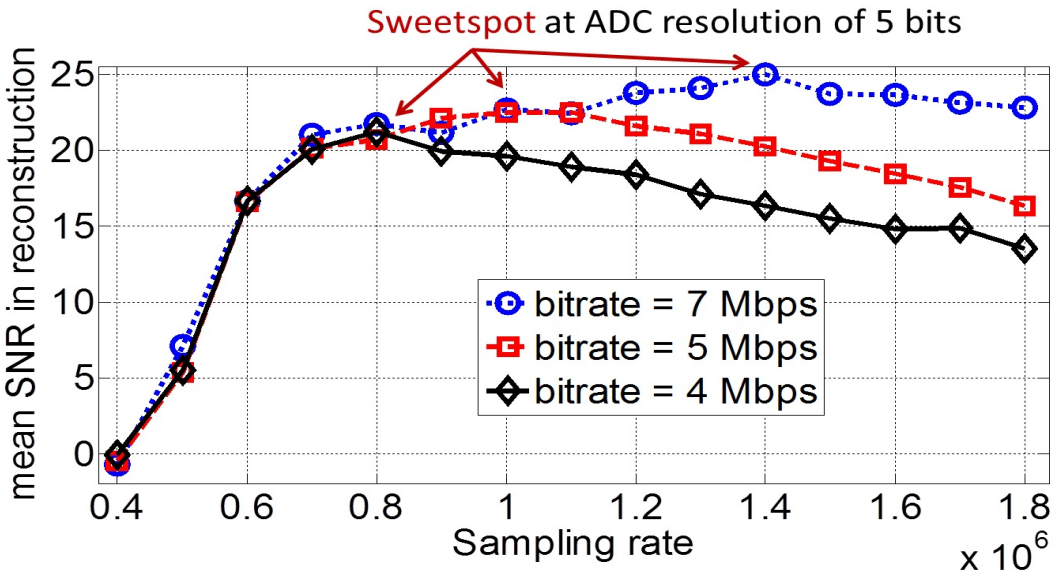

Figure 2.14: Mean output SNR versus random PPM ADC sampling rate, for fixed bitrates of $4, 5$ and 7 Mbps.

sampling rate. For example, a bit-rate of 5 Mbps can be achieved by choosing an ADC quantization of 5 bits and a sampling rate of 1 MHz. Fig. 2.14 displays the constant bitrate curves for bitrate values of $4, 5$ and 7 Mbps for a random 11-tone input signal with input SNR of 15 dB. Each curve plots the mean output SNR for varying sampling rate. A low sampling rate corresponds to high ADC resolution and vice-versa (since the bitrate is fixed for each curve). If the sampling rate is too low, resulting in a lack of enough measurements, the reconstruction error increases, degrading the output SNR. If the sampling rate is too high, the output SNR again degrades due to lack of sufficient resolution in each measurement. This trade-off results in a sweetspot where the SNR performance is the best. From Fig. 2.14, we observe that this sweetspot occurs when the ADC resolution is chosen to be about 5 bits.

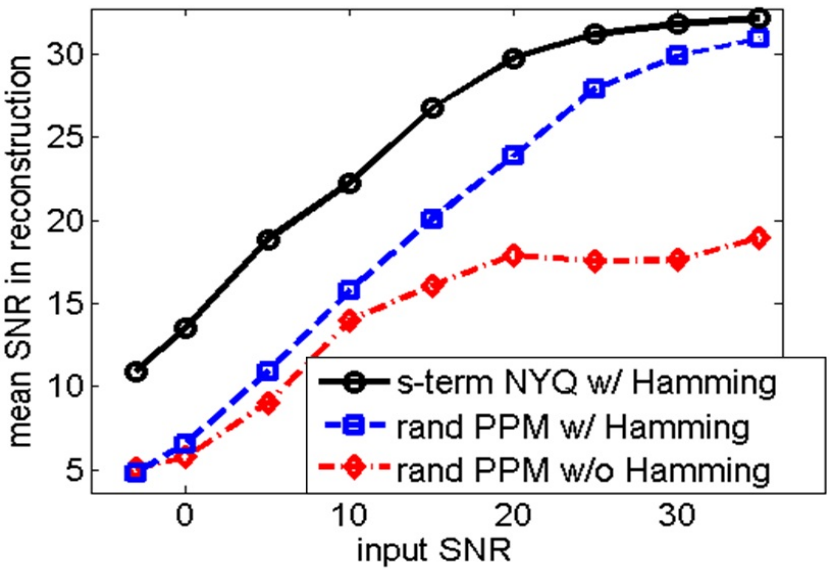

Figure 2.15: Output SNR vs input SNR for a demodulated FM signal

FM signal with off-grid frequencies If a frequency falls in between two Nyquist grid points, it causes spectral spread or leakage, thus adversely affecting the sparsity of the signal. To counter this we propose to multiply the measurements from the ADC (before reconstruction) with a window function, like the Hamming (which is non-zero at all times, hence its effect can be reversed after the reconstruction). A frequency modulated (FM) signal with single tone message, where both the carrier and message frequencies are appropriately chosen to be off-grid acts as the input signal for this experiment. At 33% sampling, the noisy FM signal is windowed, reconstructed (using PRSreco algorithm), demodulated and the resultant output SNR is plotted in Figure 2.15. Some output SNR is lost as the amplitude of the message is smaller than the dynamic range of the ADC. The use of Hamming window improves the performance of the algorithm at all SNR levels and approaches the benchmark as SNR increases. Similar observations have been made for amplitude modulated (AM) signals with different message signals. The plots exhibit similar qualitative behavior when the sampling rate is increased or decreased. Note that for an FM

signal, windowing need not be reversed as the message is in the frequency of the signal.

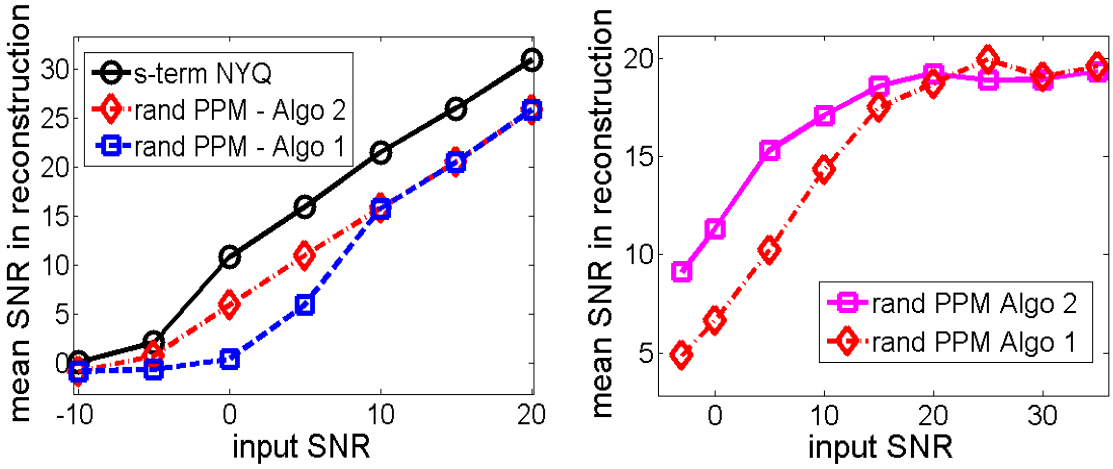

Figure 2.16: Output Vs Input SNR for a (a) multitone signal (b) demodulated AM signal with a sawtooth message

Comparison of algorithms We now repeat the multi-tone (with 13 on-grid frequencies) signal reconstruction and the reconstruction of AM signal with sawtooth message (off-grid frequencies) experiments with the MOE algorithm, choosing the number of blocks $m = 7$. From the Fig. 2.16 we see that at low SNR conditions ALGO 2 (MOE) gives a better performance than ALGO 1 (PRSreco). This is because the identification stage in ALGO 2 (MOE) is more successful as it nullifies the effect of noise to some extent by taking the median over a set of m blocks. At high SNR, both the methods give comparable performance even though ALGO 2 (MOE) has only 1 iteration. Hence ALGO 2 (MOE) can be used to reduce computations whenever the input signal satisfies the additional conditions (in section 2.8), particularly if it is also known that the input SNR levels are low. We also observed that when Hamming window is employed the performance of ALGO 1 (PRSreco) improves whereas the ALGO 2 (MOE) shows little to no improvement. This is be-

cause, upon application of the Hamming window the input signal does not strictly satisfy the assumptions made in section 2.8 and hence the improvement in sparsity of the signal is balanced by the error amplification due to Hamming window reversal.

2.9.2 Prototype and measurement results

We now present experimental results obtained with the prototype 9-bit random PPM ADC and 9-bit regular PPM ADC. The ramp generator, comparator and the two-step TDC, which are part of both the random PPM and the regular PPM ADCs, are implemented in 90 nm digital CMOS. The LFSR-based random clock generation block is implemented by programming Verilog code onto a Field-Programmable gate array (FPGA). The analog circuits operate with a 1 V supply, while the digital blocks operate at near-threshold from a 400 mV supply. The regular clock is a 64 MHz signal giving a $T_{clk} = 15.63$ nsec. The LFSR is 9 bits with taps at bin 5 and 9 resulting in a LFSR periodicity of 511. The entire evaluation setup of the random PPM ADC consists of four main blocks as displayed in Fig. 2.17, an FPGA, the ADC, a Logic Analyzer and a computer. The FPGA generates the *start* and the random clock signals, which are input to the PPM ADC. The ADC measurements are collected by the logic analyzer. The non-zero seed used to initialize the LFSR system is assumed to be known during reconstruction, so that the sequence of t_{RAND}'s can be calculated.

A single tone input signal is sampled both by the random PPM ADC prototype and the regular PPM ADC prototype, operating at various sampling rates, and reconstructed using the PRSreco algorithm. The results are displayed in Fig. 2.18. Also displayed for convenience is the compression loss (root mean square error of the reconstruction) on the right y-axis. Note that since the ADC resolution is fixed,

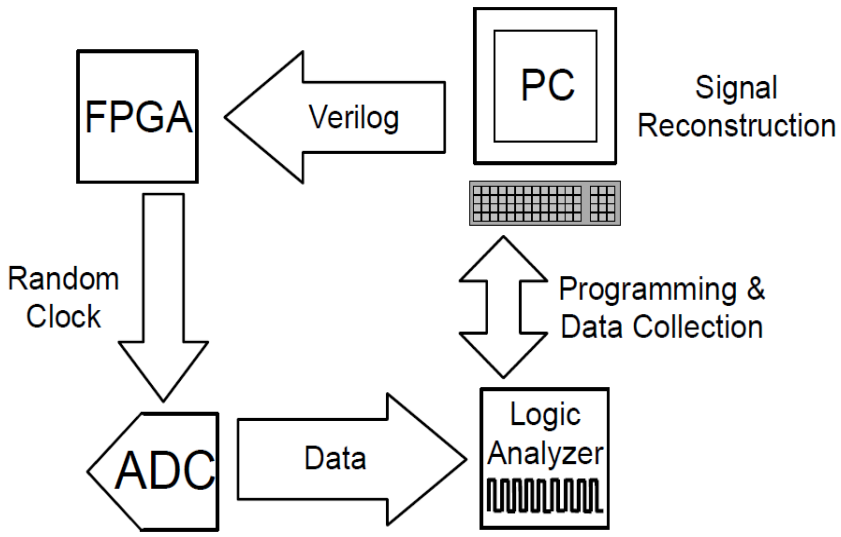

Figure 2.17: Hardware setup for the random PPM ADC

the compression achieved by the sampling scheme only depends on the sub-sampling ratio K/N. As expected, the random PPM performs much better than the regular PPM which breaks down when the sub-sampling ratio is around 0.7, whereas the random PPM works well for sub-sampling ratios as low as 0.05. A compression ratio of 0.05 in the random PPM ADC and 0.7 in the regular PPM ADC, both result in the same compression loss of 0.77. Fig. 2.19 shows the reconstruction of a 5-tone signal with frequencies arbitrarily chosen from the Nyquist grid on $[0, 1\text{MHz}]$ (Nyquist rate $= 2$ MHz). The multi-tone signal was sampled with the random PPM ADC operating at a sampling frequency of about 173 KHz which leads to a sampling percentage of about 8.65%. The SNR of the recovered signal is 41.6dB.

The measured power consumption of the PPM ADC system is 14μW (excluding digital post-processing). The analog and digital blocks each consume 7μW. For the random PPM ADC system, the expected improvement in the power by a factor of K/N (the sub-sampling ratio) is observed, however this does not include the power consumed by the random clock generator and the comparator, both of which oper-

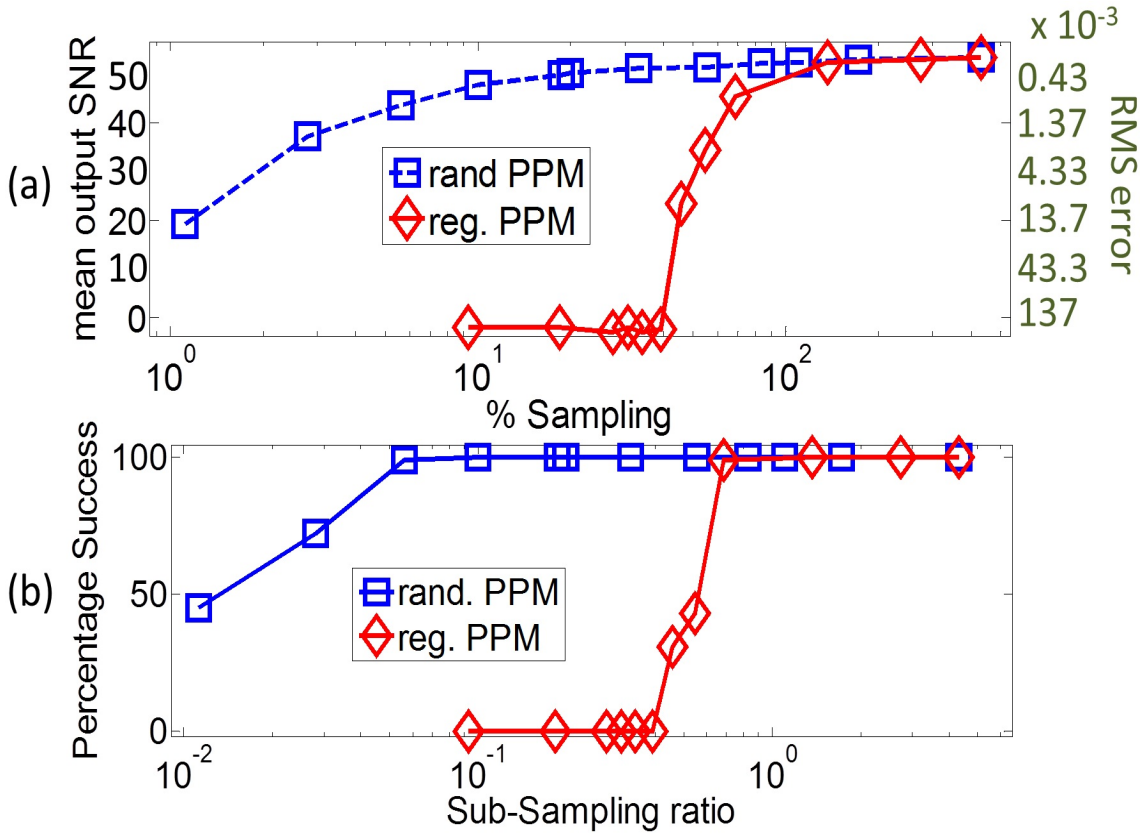

Figure 2.18: Reconstruction of a single tone signal from samples collected by the regular and the random PPM ADC prototypes operating at varying sampling rates. The y-axis on the right displays the corresponding root mean square (rms) error.

ate all the time. The random clock signal is produced by the FPGA and the power consumption of the FPGA itself is not a good indication of the actual power needed since power is consumed by unnecessary circuitry in the FPGA. Implementing the random clock generation on the CMOS IC along with the rest of the compressive sensing ADC would only minimally increase the power consumption of the IC as the LFSR system only requires on the order of ten shift registers and a few gates. The PPM ADC itself uses approximately 50 registers and gates [1], therefore, the digital power consumption due to the addition of LFSR system, is expected to increase by only $6 - 8\%$. An additional power reduction can be achieved by switching the continuous-time comparator off, when not in use. Thus, the power consumption of

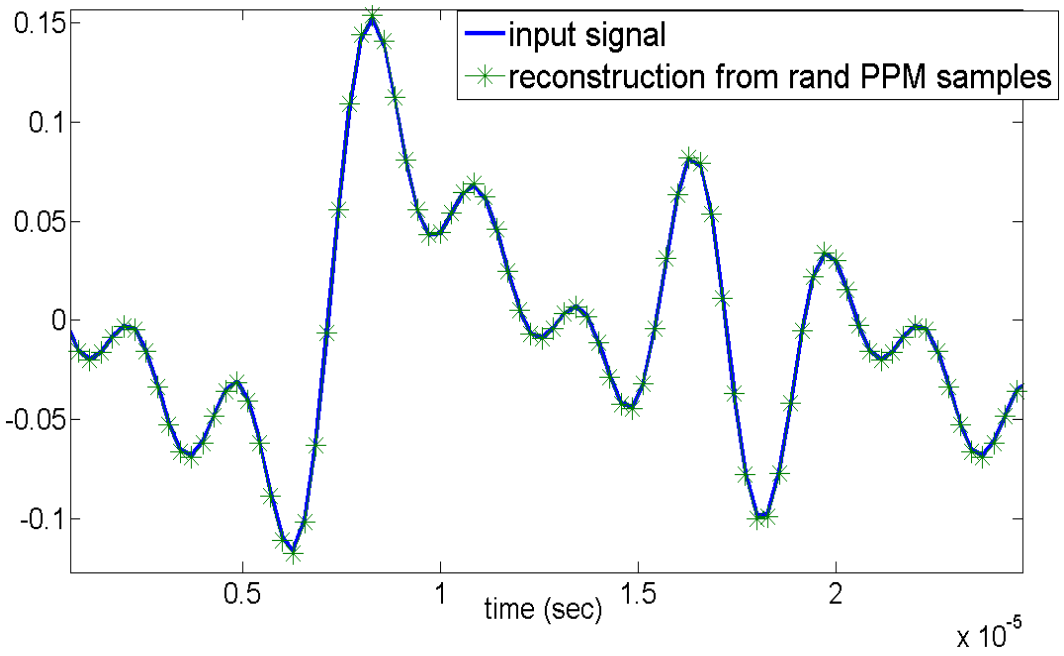

Figure 2.19: Reconstruction of a 5-tone signal from samples collected by random PPM with sampling rate at 8.65% of the Nyquist rate

the random PPM ADC (with on-chip random clock generation) is estimated to be about $(2\frac{K}{N} + 0.07)7\mu$W. For a random PPM ADC operating at 20% of the Nyquist sampling rate (= 1 MHz), the estimated power consumption is 3μW.

2.10 Conclusion

We propose a new low power compressive-sampling analog to digital converter, called the random PPM ADC. It inherits the advantages of time to digital conversion and also exploits compressive sampling techniques, to improve the power efficiency of data conversion. An existing PPM ADC design is modified to achieve a 9-bit random PPM ADC, through the use of a random clocking technique. The new random design enables the reduction of the average sampling rate to sub-Nyquist levels and thus reduces the ADC power consumption by a factor close to the sub-sampling ratio.

The random PPM performs much better than a regular PPM operating at a sub-Nyquist sampling rate, in terms of obtaining closer-to-benchmark output SNR and handling signals that are less sparse. The proposed reconstruction algorithm is not only faster (greedy pursuit versus basis pursuit inspired algorithms in the literature for compressive sampling ADCs) but also feasible for a hardware implementation. With on-chip reconstruction and a low power front-end, the random PPM ADC is attractive for power constrained applications such as wireless sensor networks, as it reduces both the power consumption and the amount of data that needs to be communicated by each sensor node.

CHAPTER III

Model Of A Sparse Encoding Neuron

3.1 Introduction

Neurons as Time Encoding Machines (TEMs) have been proposed to capture the information present in sensory stimuli and to encode it into spike trains [51, 52, 53]. These neurons, however, produce spikes at firing rates above Nyquist, which is usually much higher than the amount of information actually present in stimuli. We propose a low-rate spiking neuron which exploits the sparsity or compressibility present in natural signals to produce spikes at a firing rate proportional to the amount of information present in the signal rather than its duration. We consider the IAF (Integrate-and-Fire) neuron model, provide appropriate modifications to convert it into a low-rate encoder and develop an algorithm for reconstructing the input stimulus from the low-rate spike trains. Our simulations with frequency-sparse signals demonstrate the superior performance of the *Low-Rate* IAF neuron operating at a sub-Nyquist rate, when compared with IAF neurons proposed earlier, which operate at and above Nyquist rates.

It is a common belief that neurons encode sensory information in the form of a sequence of action potentials (nerve impulses or "spike trains"). The fundamental

unit of a "message" conveyed by a neuron is a single nerve impulse, propagating at high speed down its axon through well-understood electro-chemical processes [54]. These "spike trains" are interpreted by other neurons, leading to sensation and action. Fig. 3.1 illustrates a spike train produced by an auditory nerve cell. When we hear something, our brain is not actually interpreting the modulations in the acoustic waveform, but rather the spike trains generated, in response to the stimulus, by thousands of auditory nerves. In other words, spike trains form the language that the brain uses to communicate between neurons. Hence, understanding how a neuron encodes the stimulus or input signal into spike trains is of great interest.

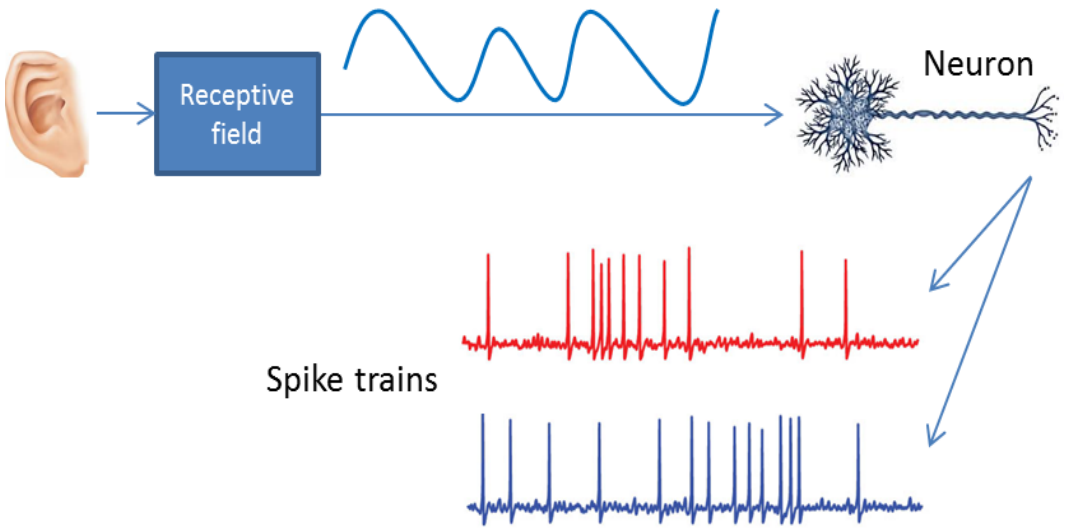

Figure 3.1: Spike trains produced by an auditory neuron

Neurons generate spikes at relatively low rates, presumably due to a metabolic reason [55]. Metabolically efficient coding [56] is indicative of sparse encoding. Further, it has been observed that the process of spike encoding exhibits variability or randomness in response to identical inputs [57]. That is, for the same input stimulus, the neuron may produce different spike trains (as shown in Fig. 3.1). We are interested in developing a sparse encoding model of neuron that explains these observed

features of spike trains.

Integrate-and-fire (or IAF, in short) models for neurons as generating time-stamp codes have been studied in [51] and [53]. Lazar et al. proved that a band-limited signal encoded by the precise spike timing of an IAF neuron can be reconstructed with reasonable accuracy from the spike train, when the average firing rate is above Nyquist rate [51]. When no other information is available about the input signal except its bandwidth, the signal has to be encoded at above Nyquist rate for successful recovery. However, most natural signals are often sparse or compressible in some orthonormal basis and hence the actual information present in the signal is usually much lower than the Nyquist rate.

From an information theoretic point of view, a sparse encoding neuron should be able to encode such signals using spike trains that have a rate proportional to the amount of information actually present in the signals. In other words, most natural signals live in a low dimensional space and an efficient encoder should be able to capture the low dimensional information from the high dimensional signal. In this paper, we develop an efficient model of a sparse encoding neuron, which we call the *Low-Rate IAF* neuron, by performing appropriate modifications to a conventional integrate-and-fire model. The Low-Rate IAF neuron exploits the sparsity or compressibility of input signals to encode them into spike trains with rates well below the Nyquist rate. We show that the low-rate spike trains contain enough information about the input stimulus to allow its recovery and develop a neural decoding algorithm based on spike times.

The remainder of the chapter is organized as follows. The input signal model is described in Section 3.2. A relevant background on time encoding through integrate-and-fire neurons, including the model proposed by Lazar in [53], is briefly presented in Section 3.3. The proposed Low-Rate IAF neuron is presented in Section 3.4 and is followed by a description of the reconstruction algorithm in Section 3.5. A set of numerical experiments compare the performance of Lazar's IAF neuron (from [53]) with the proposed Low-Rate IAF neuron in Section 3.6. We conclude with a discussion on future work in Section 3.7.

3.2 Input stimulus model

The class of input signals is assumed to be band-limited with cutoff frequency W (in Hz) and periodic within a time period D. The Nyquist rate of the input signal space is thus $F_N = 2W$. W and D are related by

$$W = \frac{N}{2D}$$

where N is a positive integer that denotes the dimension of input space. If an input signal/stimulus $x(t)$ is sampled at Nyquist rate for a time duration of D, then the number of samples obtained is $N = F_N D$. Thus the signal $x(t)$, observed for a time duration D, can be represented as a vector x of length N in discrete domain, where

$$x[i] = x(i/F_N)$$

for $i = 1, .., N$. The signal $x(t)$ is further assumed to be S-sparse or compressible in frequency domain. A signal is called S-sparse in the frequency domain, if the DFT (discrete Fourier transform) of the signal samples at Nyquist rate has only S non-zero terms. That is, if X represents the DFT of vector x, then X has at most

S non-zero elements. A signal is called S-compressible[1] in frequency domain, if the sorted list of its DFT coefficients has only S significant or dominant terms, compared to which the other terms are negligible. Thus, a compressible signal is one that is reasonably well approximated as a sparse signal.

The input signal can be expressed as a linear combination of complex exponentials as follows:

$$x(t) \cong \sum_{m=1}^{S} c_m \exp(j2\pi f_m t)$$

where $f_m, m = 1, .., S$ are the S dominant frequencies which lie in the interval $[-W, W]$ and c_m are the corresponding coefficients. We further assume that the input signal is real-valued, hence S is even and one set of frequencies are the negative of the other set. Thus, the input stimulus is a mixture of periodic waveforms, which is consistent with the brain mechanism of generating and entraining oscillations at multiple frequencies simultaneously.

3.3 Time encoding with Integrate-And-Fire Neurons

In this section we review the time encoding machine (TEM) consisting of an integrate-and-fire (IAF) neuron [51, 52, 53]. Neurons encode continuous time sensory stimuli into discrete time events, i.e. the firing of action potentials at variable time points. Time encoding is an answer to one of the key questions arising in information processing, which is, how to represent a continuous signal as a discrete sequence. In conventional sampling, a band-limited signal is represented by set of amplitude samples spaced uniformly. If the uniform spacing is chosen to satisfy the

[1] We call X, S-compressible, if it is well approximated as a S-sparse signal, $||X - X_{(S)}||_2 \leq C \cdot S^{-\alpha}$ for some constants C and $\alpha > 0$, where $X_{(S)}$ is the S-sparse signal that best approximates X.

Nyquist rate condition, the signal can be recovered perfectly, under no noise, through sinc interpolation. This is the well-known Shannon sampling theorem. In contrast, time-encoding of a real-valued band-limited signal is an asynchronous process of mapping the amplitude information into a strictly increasing sequence of time points. A time encoding machine (TEM) is a realization of such encoding. The reconstruction of input signal from the sequence of time points is referred to as time decoding.

3.3.1 Preliminaries

A typical IAF TEM neuron is schemtaically shown in Fig. 3.2. A constant bias b ($b > 0$ such that $x(t) + b > 0, \forall t$) is added to the input signal, which is then fed to the integrator. When the output of the integrator crosses a threshold δ, a spike is produced. The spike triggers a zero reset of the output of the integrator. The output of the TEM is thus a sequence of spikes at time points, $\{t_k\}$, that models the spike train produced by a neuron.

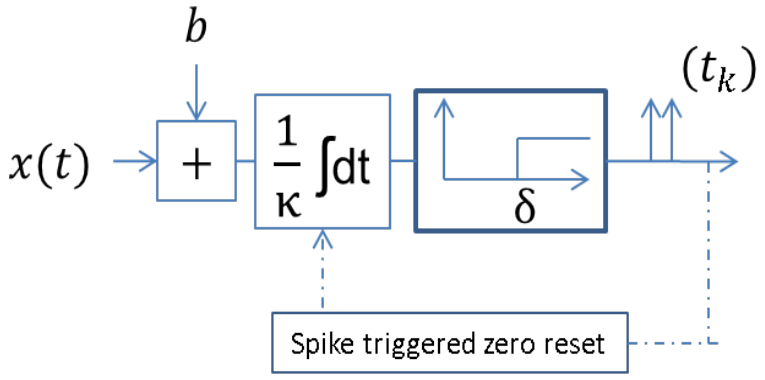

Figure 3.2: Time encoding with an integrate-and-fire (IAF) neuron

Let K denote the number of spikes produced by the IAF neuron in the duration D for which the input stimulus is observed. From simple calculations we can easily

derive,

$$\int_{t_k}^{t_{k+1}} x(s)\,\mathrm{d}s = \kappa\delta - b(t_{k+1} - t_k)$$

for $k = 0, .., K-1$, where t_0 is the time point at which we begin to observe the signal. If $|x(t)| \leq c$, $\forall t$, then the inter-spike-interval is bounded by,

$$\frac{\kappa\delta_k}{b+c} \leq t_{k+1} - t_k \leq \frac{\kappa\delta}{b-c}$$

It has been proved [51][52] that a successful recovery of x is possible when,

$$\frac{\kappa\delta}{b-c} < \frac{1}{2W}$$

that is, the maximum inter-spike-interval is smaller than the Nyquist period $T_N = 1/F_N = 1/2W$. Hence, the TEM IAF neurons encode all input signals at an average rate greater than the corresponding Nyquist rate.

3.3.2 Integrate-And-Fire Neurons with Random Thresholds

To model the variability or randomness characteristic of neuronal spike trains, neurons with random thresholds were proposed in [58]. An IAF neuron model with random thresholds is studied by Lazar in [53]. The model is identical to the TEM shown in Fig. 3.2, but with random thresholds δ_k. Every output spike not only resets the integrator output but also triggers the random selection of a new threshold δ_k. The random thresholds are assumed to be drawn from a Gaussian distribution with known mean δ and variance σ^2.

For random thresholds TEM, let us define a measurement vector q and error vector ε of length K, as follows. For $k = 0, .., K-1$,

$$q_k = \kappa\delta - b(t_{k+1} - t_k),$$

$$\varepsilon_k = \kappa(\delta_k - \delta).$$

Time-encoding can be expressed as the following system of equations,

$$GX = q + \varepsilon$$

where X is the N-point DFT of vector x and G (of size K x N) is given as

$$G_{k,n} = \int_{t_k}^{t_{k+1}} e^{j2\pi \frac{n}{N} F_N s} \, \mathrm{d}s$$

for $k = 0, .., K-1$ and $n = -N/2, .., N/2$ (assuming N is even and with a slight abuse of notation).

A weighted least squares with ℓ_2 penalty is used for reconstructing an approximation $\tilde{X}$ of X from q

$$\tilde{X} = \text{argmin}||q - GX||^2 + K\lambda||X||^2.$$

Here, λ is a positive smoothing parameter that regulates the trade-off between faithfulness to the measurements and smoothness. The regularization is used to prevent over-fitting due to the noisy data.

For a successful recovery, the method requires that the average spike rate be above Nyquist rate [53]. In other words, we need the number of spikes $K > N$. Note that N is the number of samples at Nyquist rate and hence Lazar's TEM neuron is firing at rates above Nyquist. In the next section we develop a low-rate model of IAF neuron that fires at a sub-Nyquist rate.

3.4 The Low-Rate Integrate-and-Fire Neuron

We introduce appropriate modifications in the TEM IAF neuron and develop a low-rate IAF neuron model. The *Low-Rate* IAF neuron schematical is shown in Fig. 3.3. We use fixed thresholds (δ) as opposed to random thresholds used in Lazar's model. The randomness in inter-spike-interval exhibited in neuronal spike trains is produced by an additional component that switches off the IAF circuit for a random amount of time τ_k (with mean μ) after each spike (see Fig. 3.3). The process of switching off the IAF circuit mimics the "absolute refractory" period exhibited by a neuron. After a single impulse, a dormant period occurs during which no other impulse can be initiated [54], which is called the "refractory" period. We model this refractory period as a random variable to account for the randomness in neuronal spike trains in response to identical inputs.

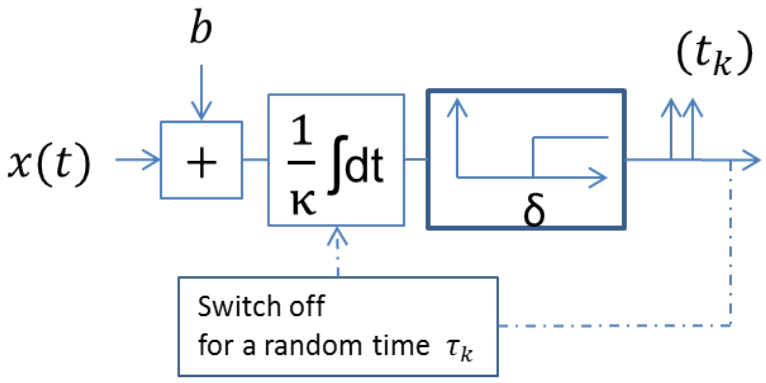

Figure 3.3: Sparse time encoding with Low-Rate integrate-and-fire(IAF) neuron.

The time durations τ_k are assumed to be uniformly distributed with mean μ. The operational equation of time-encoding can be obtained as follows,

$$\int_{t_k+\tau_k}^{t_{k+1}} x(s)\,\mathrm{d}s = \kappa\delta - b(t_{k+1} - t_k - \tau_k)$$

for $k = 0, .., K-1$. Similar to Section 3.3, we define measurement vector q and

matrix G as follows,

$$q_k = \kappa\delta - b(t_{k+1} - t_k - \tau_k)$$

$$G_{k,n} = \int_{t_k+\tau_k}^{t_{k+1}} e^{j2\pi\frac{n}{N}F_N s}\ \mathrm{d}s$$

for $k = 0, .., K-1$ and $n = -N/2, .., N/2$.

In an actual implementation of the Low-Rate IAF neuron in hardware, the time durations τ_k can be generated using a pseudo-random number generator such as linear feedback shift register (LFSR). If the seed that is used to initialize the LFSR is assumed to be known, then τ_k can be computed by the reconstruction algorithm. An alternative is to actually measure τ_k using a time to digital converter (TDC). The measurements q_k can thus be computed by the reconstruction algorithm.

The low-rate neuron produces spikes at a sub-Nyquist rate determined by the parameters δ and μ. Let K denote the number of spikes produced in duration D, then $K < N$. We are interested in solving for X (the N-point DFT of input signal) given t_k for $k = 0, .., K-1$, i.e., we want to solve the following linear system of equations for the case when $K < N$,

$$GX = q + \xi$$

where ξ is a noise vector, which can model additive noise at the input or a time jitter noise in measuring t_k. The problem is ill-posed in general, since it is underdetermined and has infinitely many solutions. But under the assumption that X is sparse or compressible (as described in Section 3.2), it may be possible to uniquely recover X. We develop a new recovery technique to reconstruct X, which is described in the next section.

3.5 The Reconstruction Algorithm

Given the measurements $q = GX + \xi$ of a sparse or compressible signal X (of length N), with number of measurements $K < N$, the novel area of *Compressive Sensing* (CS) offers explicit constructions or distributions on matrices G and algorithms such as those proposed in [8],[7] and [3], to recover an approximation of X (denoted by $\tilde{X}$). One line of research assumes that the measurement matrix G satisfies a property called the restricted isometry (RIP) [3], and uses either greedy iterative algorithms ([8],[7],[6]) or convex optimizations to obtain $\tilde{X}$. Another line of research designs matrices G and algorithms jointly, optimizing for reconstruction time [13], storage requirements of G, or physical realizability [16] of measurement with matrix G. The matrix G produced by an IAF time-encoding system (whether deterministic or random) does not necessarily satisfy the RIP condition. Hence, following the second line of research, we co-designed the measurement system (i.e. the Low-Rate IAF neuron model) and the recovery algorithm, keeping in mind the physical realizability of the TEM as well as the TDM (time decoding machine). In this section, we describe the reconstruction algorithm developed for the Low-Rate IAF neuron model presented in Section 3.4. We begin by transforming $GX = q$ into a new system of equations $BX = y$ by doing the following.

From mean value theorem, we know that there exists $s_k \in (t_k + \tau_k, t_{k+1})$ such that

$$x(s_k)(t_{k+1} - t_k - \tau_k) = \int_{t_k+\tau_k}^{t_{k+1}} x(s)\,\mathrm{d}s.$$

Thus we can define s_k for $k = 0, .., K-1$ and the corresponding signal amplitudes

as

$$y_k = x(s_k) = \frac{q_k}{(t_{k+1} - t_k - \tau_k)}.$$

We define a new measurement vector y in this manner. The N-point DFT X and measurement vector y can be related as

$$BX = y,$$

where the new measurement matrix B (of size K x N) is given by

$$B_{k,n} = e^{j2\pi \frac{n}{N} F_N s_k}$$

for $k = 0, .., K-1$ and $n = -N/2, .., N/2$. Note that B is not really a sub-DFT matrix, since $s'_k s$ do not have to lie on a Nyquist time grid.

A pseudo-code of the reconstruction algorithm is presented in Table 3.1. For a vector z, supp(z) is defined as the set of indices of the non-zero elements of z and $z_{(s)}$ stands for the best s-term approximation[2] of z. For an index set $T \subset \{1, 2, .., N\}$, z_T stands for a sub-vector of z containing only those elements of z that are indexed by T. Similarly G_T stands for a sub-matrix of G containing only the columns of G indexed by T.

The matrix B is similar to the matrix used in [16] and hence we use the algorithm developed in [16] to estimate the indices of the dominant terms in X, that is, we identify the dominant frequencies in X. The largest components in $B^H y$ provide a good indication of the largest components in X [16]. The algorithm applies this idea iteratively to reconstruct an approximation to the signal X. At each iteration,

[2]The best s-term approximation of a vector z can be obtained by equating all the elements of z to zero, except the elements that have the top s magnitudes.

the current approximation induces a residual, which is the part of the signal that has not been approximated yet. The current approximation vector $\tilde{X}$ is initialized to a zero vector and the residual is initialized to the measurement vector y. At the end of iterations, once the dominant frequencies are identified (denoted by index set T in Table 3.1), their coefficients (i.e. the elements of X_T) are then estimated through performing a least squares with a truncated matrix G_T. We approximate $s_k = (t_{k+1} + t_k + \tau_k)/2$.

The reconstruction algorithm
input: N (signal length), S (sparsity), (s_k, y_k),$k = 0, 1, .., K-1$.
output: $\tilde{X}$ (S-sparse approximation to X, length N)
$\tilde{X}^{(0)} = \underline{0}$, residual $r^{(0)} = y$ for $i = 0, 1, 2, ..$ $\quad \tilde{X}^{(i+1)} = [\tilde{X}^{(i)} + B^H r^{(i)}]_{(S)}$ $\quad r^{(i+1)} = y - B\tilde{X}^{(i+1)}$ until $\|\|r^{(i+1)}\|\|_2$ does not vary within a tolerance θ. $T = \text{supp}\left\{\tilde{X}\right\}$ $\tilde{X}_T = \left(G_T^H G_T\right)^{-1} G_T^H y$ (Least Squares) $\tilde{X}_{T^c} = 0$

Table 3.1: The Reconstruction Algorithm

The computationally intensive step of least squares is performed only once in the algorithm. The least squares is implemented using the accelerated Richardson iteration [48] with runtime of $O(SK\log(2/e_t))$ where e_t is a tolerance parameter. The structure of the measurement matrix lends us to use the inverse NUFFT [49] with cardinal B-spline interpolation for forming the products of the form $B^H r$, in a runtime of $O(N\log N)$. Hence the total runtime of the algorithm is dominated by $O(IN\log N)$ where I is the number of iterations which has a gross upper bound of

log$N$. In practice, we find that the approximation $s_k \cong (t_{k+1} + t_k + \tau_k)/2$ is good when the threshold δ is small enough. It is possible to update s_k, $k = 0, .., K-1$ using the current approximation $\tilde{X}$ at the end of each iteration, by using Newton's method for example. More sophisticated methods might yield better results.

3.6 Results and Discussion

Lazar's TEM neuron and our Low-Rate IAF neuron are simulated in MATLAB, along with the reconstruction algorithms. We compared the performance of our Low-Rate neuron firing at sub-Nyquist spike-rate with TEM neurons in [53] operating at and above Nyquist rate. We define the sparse-encoding ratio of Low-Rate IAF neuron as $\frac{K}{N}$, which implies that the firing rate of the neuron is $\frac{K}{N}F_N$. The input signal, as explained, is assumed to be a mixture of sinusoidal waveforms of S frequencies. Because we inject additive white Gaussian noise into the input signal, we use the traditional measure of signal-to-noise ratio (SNR) as the performance metric. The output SNR[3] is defined as the ratio between the signal energy and the reconstruction error, whereas the input SNR is defined as the ratio between signal energy and noise energy.

In the first experiment, we choose $S = 10$ and compare the recovery performance of Lazar's TEM neuron and Low-Rate IAF neuron. The sparse-encoding ratio of Low-Rate neuron is chosen as $K/N = 0.3052$. Fig. 3.4 plots the mean output SNR vs. input SNR. We see that the Low-Rate IAF neuron (even when operating at about one third the Nyquist rate in this example) outperforms the TEM neurons

[3]Output SNR(dB) $= 20\log(||X||_2/||X - \tilde{X}||_2)$, where X is the input signal and $\tilde{X}$ is the output of the algorithm

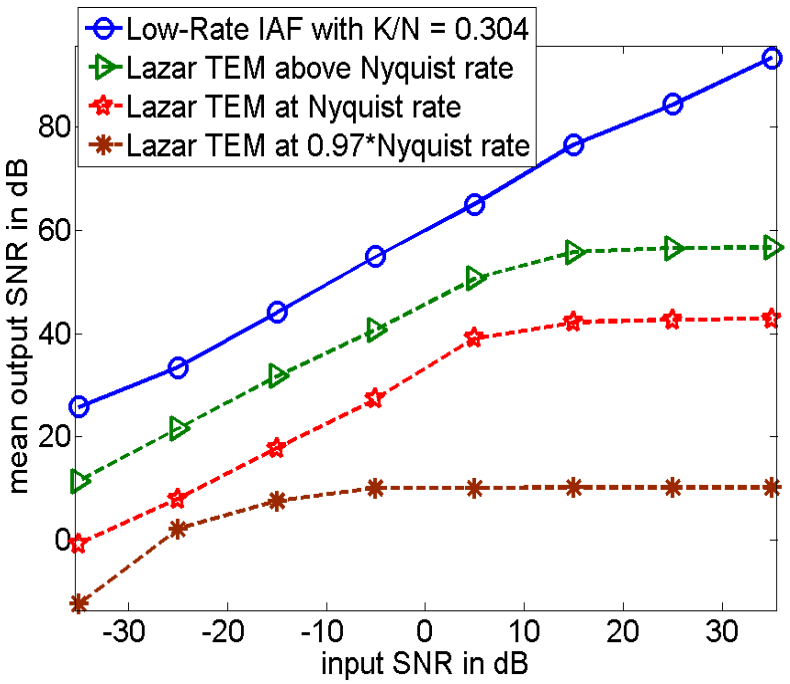

Figure 3.4: Output SNR vs input SNR for signals with $S = 10$

(which are not sparse encoders) operating at and above Nyquist rates. Moreover, we see that Lazar's reconstruction degrades significantly when the average firing rate of TEM neurons is reduced to about $0.97F_N$.

In the next experiment, we choose $S = 60$. Mean output SNR vs. input SNR is plotted in Fig. 3.5 for Low-Rate IAF neuron operating at different rates and Lazar's TEM neuron operating at about twice the Nyquist rate. To match the performance of Lazar's TEM neuron at twice the Nyquist rate, we need to set the firing rate of the Low-Rate IAF neuron to about 0.38 times the Nyquist rate. Fig. 3.5 demonstrates that an increase in sparse-encoding ratio K/N improves the performance of the Low-Rate IAF neuron.

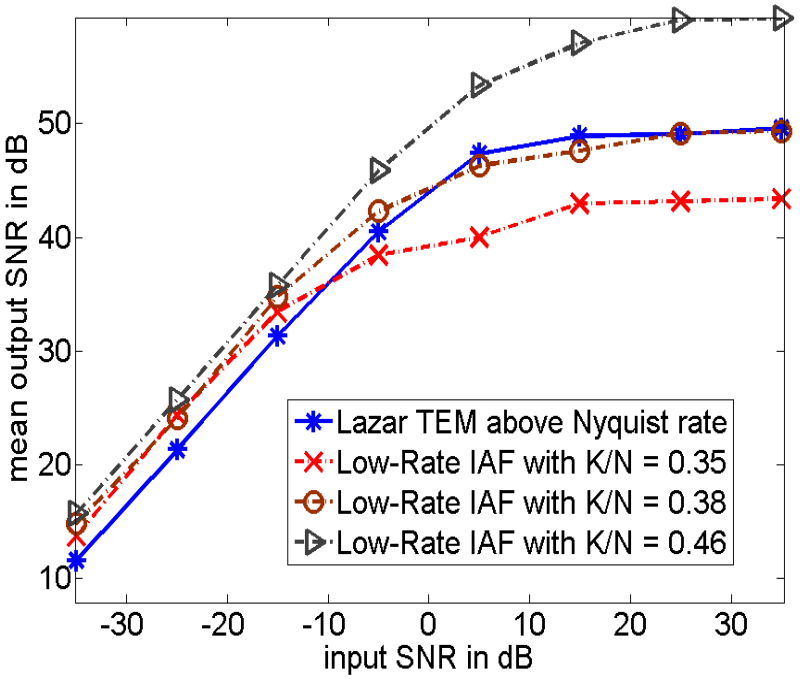

Figure 3.5: Output SNR vs input SNR for signals with $S = 60$

3.7 Conclusion and Future Work

We proposed a model for a sparse encoding neuron, called the Low-Rate IAF (integrate-and-fire) neuron, which is an adaptation of the TEM IAF model proposed by Lazar [51, 52, 53]. Lazar's TEM model produces spikes above Nyquist rate, which is usually much higher than the amount of information actually present in the input sensory stimuli. By exploiting the sparsity, the Low-Rate IAF neuron encodes input stimulus into spike trains with average firing rate well below Nyquist rate, while using the spike timing information in a smart manner to improve the performance of stimulus recovery. The developed reconstruction algorithm is computationally efficient and can be tailored for practical hardware implementations. A number of other time-encoding neuron models, including many other IAF architectures, have been proposed in the literature. The methodology of low-rate or sparse encoding, along with the developed reconstruction algorithm, can be extended to these neuron

models. This direction will be explored in the future. We are also interested in investigating the application of our Low-Rate neurons in developing a sparse encoding model for videos. The classification of input stimuli from low-rate spike trains is another potential future direction.

CHAPTER IV

Continuous Fast Fourier Sampling

4.1 Introduction

The problem of quickly computing the largest few Fourier coefficients of a signal from a given (sliding) time window arises in numerous situations. For example, in *cognitive radio* [59], where a wireless node alters its transmission or reception parameters based on active monitoring of radio frequency spectrum at various times, or in *incoherent demodulation* of communication signals [60] (such as FSK, MSK, OOK, etc.,) where the computed frequency spectrum at different times represents the message being transmitted itself. Other applications include data compression, feature extraction, data mining, continuous monitoring of signals, real-time change detection in signal parameters, etc. Most of these applications involve large signal sizes or bandwidths while the signal is often redundant (sparse or compressible), with only a few Fourier coefficients that are of interest. In such cases the Fast Fourier Transform (FFT), which computes all the Fourier Transform (FT) terms, is computationally wasteful. Hence algorithms with efficient storage requirements and low runtime are of primary importance. Moreover, the resource efficient algorithm should be able to quickly analyze a signal from any arbitrary placed time window.

Compressed Sensing (CS) methods [3] [6] provide a robust framework for reducing the number of signal samples required to estimate a signal's Fourier transform. Although the storage requirements are small, standard CS Fourier methods often utilize Basis Pursuit (BP) [3] and greedy matching pursuit algorithms [6] that have a runtime super-linear in signal's size/bandwidth, and hence, inappropriate for applications such as those described above. A second body of work on algorithmic compressed sensing includes methods which focus on achieving near-optimal running times [61] [62]. However these algorithms do not achieve sub-linear storage requirements.

Fourier sampling algorithms [11] [63] achieve both sub-linear storage and runtime requirements by exploiting the spectral redundancy of signals. In particular, a randomized Fourier sampling algorithm called the AAFFT (Ann Arbor Fast Fourier Transform) [63] has been shown to outperform the FFT in terms of runtime while utilizing only a fraction of the FFTs required samples [64]. In these algorithms, unevenly spaced samples of the signal (from a given time window) are acquired in a structured random fashion, below Nyquist rate. These samples are used in a non-linear iterative manner to quickly estimate the signal's dominant Fourier coefficients. The structure in the random sampling pattern, however, depends upon the boundaries of the time window in which the signal is analyzed (see Section 4.2.2). Thus an arbitrary placing of the analysis window is not accommodated. We propose the Continuous Fast Fourier Sampling (CFFS) algorithm which is both a highly efficient reconstruction algorithm (like AAFFT) and adapted for arbitrary sliding window calculations, thus attractive for the mentioned applications of interest.

The AAFFT algorithm and its limitations are briefly discussed in Section 4.2.2. The CFFS algorithm is described in detail in Section 4.3, followed by theorems that prove its correctness in Section 4.3.3. Section 4.4 presents a few results and numerical experiments that provide proof of concept and apply the CFFS algorithm to decoding frequency hopping signals with known and unknown change points.

4.2 Background and preliminaries

The algorithms in this chapter and their analysis are inherently discrete. The samples are drawn from a discrete time signal (rather than an underlying continuous-time signal) and output of the algorithms is an approximation to the discrete Fourier spectrum of the signal.

4.2.1 The problem setup and notation

Let the input discrete time signal be denoted by x of length n (n very large). Let y denote the signal x from a given analysis window or block of length N ($N << n$ and $N = 2^{\alpha}$ for some integer α). If (n_1, n_2) are the boundaries of the analysis window, then $n_2 - n_1 + 1 = N$ and $y(i) = x(i - n_1 + 1)$, for $i = 1, .., N$. y is assumed to be sparse or compressible in the frequency domain. A signal is called m-sparse in frequency domain, if its Discrete Fourier transform (DFT) has only m non-zero terms, while it is called m-compressible in frequency domain, if the DFT has m dominant coefficients with other negligible coefficients. So y can be viewed as superposition of m dominant frequencies. An algorithm is called sub-linear if it has O(m poly(log(N)) runtime and storage requirements. Furthermore, an algorithm is called "continuous" or "sliding window algorithm" if it can accommodate arbitrary positions of block y.

We develop the CFFS algorithm (section 4.3), which is a sub-linear sliding window algorithm.

4.2.2 The Ann Arbor Fast Fourier Transform (AAFFT)

The AAFFT is predicated upon non-evenly spaced samples (from block y), unlike many traditional spectral estimation techniques [65, 66] and uses a highly nonlinear reconstruction method that is divided into two stages, *frequency identification* of the m dominant frequencies and *coefficient estimation*, each of which include multiple repetitions of basic subroutines. A detailed description of the implementation of AAFFT is available in [63].

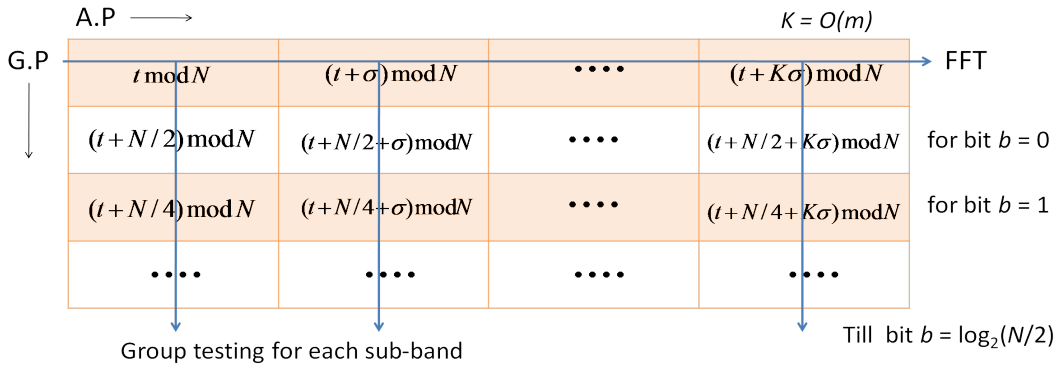

Figure 4.1: Figure showing the samples acquired in AAFFT for each (t, σ) pair

Frequency Identification consists of two steps, dominant frequency isolation and identification. Isolation is carried out by a two-stage process: (i) random time dilation of y (corresponds to a random permutation of the spectrum of y), followed by (ii) the application of a filter bank with $K = O(m)$ filters. The probability of isolation of dominant frequencies by different filters is increased with repetitions. Note that all the above is carried out *conceptually* in the frequency domain but instantiated

in the time domain. In each repetition, a pair (t, σ) is chosen randomly with $t \sim$ U$[1, 2, .., N]$ and $\sigma \sim$ U$[1, 3, .., N-1]$ and the samples of the signal block y indexed by the matrix in Fig. 4.1 are used to perform computations. Let $P(t, \sigma) = \{(t + q\sigma) \mod N, q = 0, 1, .., K-1\}$ be the arithmetic progression that forms the first row in figure 4.1. The other rows consist of arithmetic progressions $P(t^b, \sigma)$, where t^b is an element of the geometric progression $t^b = t + \frac{N}{2^{b+1}}, b = 0, 1, .., \alpha - 1$. The isolation stage performs K-point FFT along each row of the matrix. After the FFTs, the i^{th} column contains the output of i^{th} filter in the bank, evaluated at time points $t, t + N/2, t + N/4, ...$ given by the above geometric progression. The identification stage performs group testing across each column to determine the (bits of the binary representation of the) dominant frequency isolated by the corresponding filter. Let $A_1 = \{(t, \sigma)\}$ be the set containing all the (t, σ) pairs used in the *frequency identification* stage. Similarly, let A_2 be the set containing the (t, σ) pairs used in the *estimation* stage (which also uses the random sampling pattern similar to the first row of figure 4.1, for coefficient estimation of each of the identified dominant frequencies). The whole process takes time and storage in the order of $m\text{poly}(\log(N))$.

Note that although the (t, σ) pairs in A_1 and A_2 are chosen randomly, the sample indices that result from each pair are highly structured. Moreover, the indices are dependent on the boundaries of block y (due to the $\mod N$ arithmetic). Thus AAFFT can analyze the input signal x by dividing it into consecutive non-overlapping blocks or windows of length N. Let us call this block-based analysis method $S1$. $S1$-AAFFT clearly cannot accommodate arbitrary position of analysis window. This is illustrated in Fig. 4.2 for a simple case of $N = 32$ with a dummy y-axis for clarity. The X's represent the indices where the samples are acquired by the $S1$-AAFFT

procedure from two consecutive blocks $B1$ and $B2$. The O's represent the indices where the samples are needed for applying AAFFT to an arbitrarily chosen block B. As can be seen in the figure, the $S1$-AAFFT procedure did not acquire all the O's.

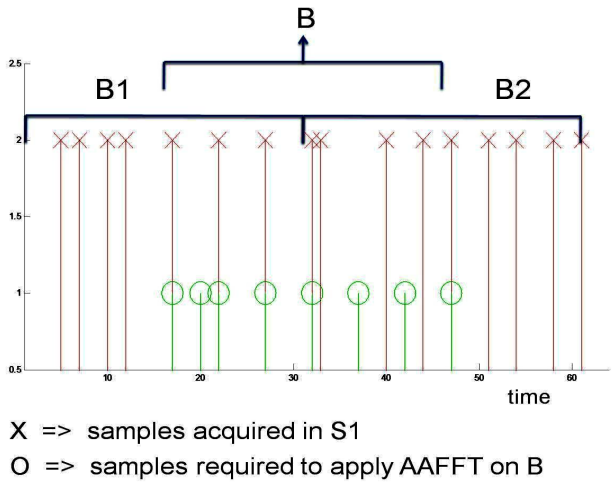

Figure 4.2: Figure showing the samples acquired by $S1$ (X's) and the samples (O's) required to apply AAFFT on $B = [16, 47]$

4.3 Continuous Fast Fourier Sampling

In this section we construct a new sampling procedure for signal x, called the CFFS, that permits a fast reconstruction algorithm (like AAFFT) on arbitrarily placed analysis windows of length N from signal x.

4.3.1 Sample set construction

For each (t, σ) pair, define a sequence of time points $t(j), j = 1, .., J$ (with $t(0) = t$ and $J = \lceil \frac{K\sigma}{N} \rceil$) such that $t(j)$ is the "N-wraparound" of $t(j-1)$. Figure (4.3) illustrates the calculation of a N-wraparound. Mathematically, $t(j) = (t(j-1) + Q(j-1)\sigma) \mathrm{mod} N$ where $Q(j-1)$ is the smallest integer such that

$t(j-1) + Q(j-1)\sigma \geq N.$

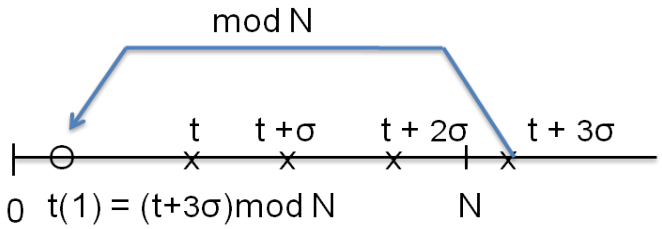

Figure 4.3: Calculation of N-Wraparound $t(1)$ from t

For $j = 1, .., J$, denote by I_j the following arithmetic progression formed by $(t(j), \sigma)$,

$$I_j = \{t(j) + q\sigma, \forall q \geq 0 : t(j) + q\sigma \leq n\} \tag{4.1}$$

Now, consider the geometric progression $t^b = t + \frac{N}{2^{b+1}}$ for $b = 0, 1, .., \alpha - 1$. For each b, $\left(t + \frac{N}{2^{b+1}}, \sigma\right)$ is treated as another (t, σ) pair and the sequence $t^b(j)$ and the corresponding progressions I_j^b can be calculated. For each pair (t_ℓ, σ_ℓ) in A_1 and A_2, expand as above and denote the arithmetic progressions produced by $I_{\ell,j}$, for $j = 1, .., J_\ell$. Define the union of all such arithmetic progressions as $I_\ell = \bigcup_{j=0}^{J_\ell} I_{\ell,j}$. I_ℓ is shown in Fig. 4.4.

$t_l = t_l(0)$	$t_l(0)+\sigma_l$	••••	$t_l(0)+q\sigma_l \leq n$
$t_l(1)$	$t_l(1)+\sigma_l$	••••	$t_l(1)+q\sigma_l \leq n$
••••	••••	••••	••••
$t_l(J_l)$	$t_l(J_l)+\sigma_l$	••••	$t_l(J_l)+q\sigma_l \leq n$

Figure 4.4: Figure showing the arithmetic progression samples acquired in CFFS for a (t_ℓ, σ_ℓ) pair and their wraparounds

Similarly define $I_\ell^b = \bigcup_{j=0}^{J_\ell} I_{\ell,j}^b$ for each $b = 0, .., \alpha - 1$.

Now define the union $I_\ell^B = \bigcup_{b=0}^{\alpha-1} I_\ell^b$. Finally define

$$I(A_1, A_2) = \left(\bigcup_{A_1} (I_\ell \cup I_\ell^B) \right) \cup \left(\bigcup_{A_2} I_\ell \right). \tag{4.2}$$

Given a set of indices I, we denote by $S^x(I)$ the set of samples from signal x indexed by I.

4.3.2 The CFFS Algorithm

Preprocessing
input: N // Block length (1) Sample-set generation : Choose A_1 and A_2 as defined and compute $I(A_1, A_2)$ (as in Equation (4.2)). **output:** $I(A_1, A_2)$ // Index set
Sample Acquisition
input: $I(A_1, A_2)$, x (2) sample signal x at I and obtain samples $S^x(I)$. **output:** $S^x(I)$
Reconstruction
input: $S^x(I)$, (n_1, n_2) // boundary indices of an arbitrary block y of length N from signal x (3) calculate A_1', A_2' (defined in Section (4.3.3)) and extract $S^y(I(A_1', A_2')) \subset S^x(I)$. (4) apply AAFFT on the sample-set $S^y(I(A_1', A_2'))$ **output:** top m frequencies of x in block $y = x[n_1, n_2]$

Table 4.1: The Continuous Fast Fourier Sampling (CFFS) algorithm

4.3.3 Proof of Correctness of CFFS

In this section we show that CFFS permits application of AAFFT on any arbitrarily placed block y in signal x. We define new sets A_1' and A_2' as follows. Put $A_1' = \{(t', \sigma) : (t, \sigma) \in A_1\}$, where t' is the n_1-wraparound of t. Mathematically, $t' = (t + i\sigma) \mathrm{mod} n_1$ where i is the smallest integer such that $t + i\sigma > n_1$. Similarly define A_2'. Note that A_1' and A_2' are still random since A_1 and A_2 were chosen randomly. AAFFT is applied on y with the sampling pattern defined (in Section (4.2.2)) from A_1' and A_2'. The following theorems together show that the required samples of y are available in $S^x(I(A_1, A_2))$.

Theorem IV.1. *For sets A_1' and A_2' as defined above, $S^y(I(A_1', A_2')) \subset S^x(I(A_1, A_2))$.*

Theorem IV.2. *AAFFT can be applied using the sample-set $S^y(I(A_1', A_2'))$, i.e. the index set $I(A_1', A_2')$ has the required structure explained in Section (4.2.2).*

Rather than giving detailed proofs, we prove a proposition that lies at the heart of the two theorems.

Proposition IV.3. *For every (t', σ) in A_1' or A_2', $S^y(P(t', \sigma)) \subset S^x(I(A_1, A_2))$.*

Proof. Let (t, σ) be the pair in A_1 or A_2 from which (t', σ) was obtained. We will prove that the arithmetic progressions I_j formed by the sequence of wraparounds $t(j)$,$j = 1, .., J$ as defined in Section (4.3.1), induce mod-N arithmetic in the progression $P(t', \sigma)$ (P as defined in Section (4.2.2)). Consider the first few terms in $P(t', \sigma)$, till $(t' + (q_0 - 1)\sigma)$ mod N where q_0 is the smallest integer such that $(t' + q_0\sigma) \geq N$. From definition of t' observe that $t' = (t + i\sigma - n_1)$. so

$$y(t') = x(n_1 + t') = x(t + \sigma) \in S^x(I_0),$$

where I_0 is defined in Equation (4.1). Similarly it is easy to see that the first q_0 terms in $S^y(P(t', \sigma))$ are contained in $S^x(I_0)$. Now call the next term $(t' + q_0\sigma)$ mod $N = t'(1)$. Observe that $t'(1) = t' + \sigma \left\lceil \frac{N-t'}{\sigma} \right\rceil - N$. Similarly observe that $t(1) = t + \sigma \left\lceil \frac{N-t}{\sigma} \right\rceil - N$. Now, Substituting $t' = (t + i\sigma - n_1)$ in the expression for $t'(1)$ we get,

$$\begin{aligned} t'(1) &= t + i\sigma - n_1 + \sigma \left\lceil \frac{N - t + n_1 - i\sigma}{\sigma} \right\rceil - N \\ &= t + i\sigma - n_1 + \sigma \left\lceil \frac{N - t}{\sigma} \right\rceil + d\sigma - N \\ &= t(1) + (i + d)\sigma - n_1, \end{aligned}$$

for an appropriately defined d, which can be shown to be positive. So,

$$\begin{aligned} y((t' + q_0\sigma) \bmod N) &= y(t'(1)) \\ &= x(t(1) + (i + d)\sigma) \\ &\in S^x(I_1), \end{aligned}$$

where again I_1 is defined in Equation (4.1). Let q_1 be the smallest integer such that $(t'(1) + q_1\sigma) \geq N$. Now it is easy to see that the next q_1 terms in $S^y(P(t', \sigma))$ are contained in $S^x(I_1)$. Repeat this until all the terms in $P(t', \sigma)$ are covered.

□

Proposition IV.4. *On average, the storage requirement of CFFS algorithm is in the order of* $O(\frac{n}{N} m \log^{O(1)} N)$, *which is of the same order as that of a sampling scheme which divides the signal into* n/N *non-overlapping blocks and samples each block for AAFFT.*

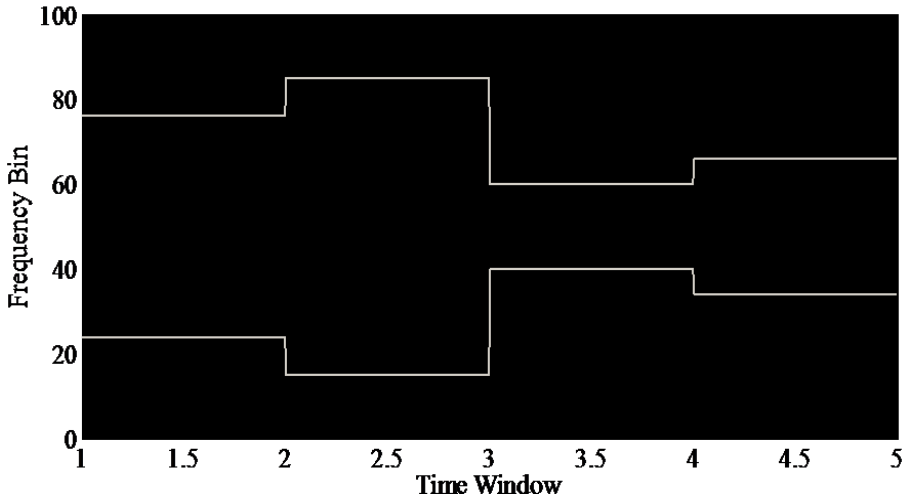

Figure 4.5: The Sparsogram (time-frequency plot that displays the dominant frequencies) for a synthetic frequency-hopping signal consisting of two tones. The same sparsogram is obtained both by AAFFT ($S1$) and CFFS

4.4 Results and Discussion

The Continuous Fast Fourier Sampling algorithm has been implemented and tested in various settings. In particular, we performed the following experiments.

Frequency hopping signal with known block boundaries: we consider a model problem for communication devices which use frequency-hopping modulation schemes. The signal we want to reconstruct has two tones that change at regular intervals which are assumed to be known. The signal is assumed to be noiseless. We apply both the straightforward $S1$-AAFFT and CFFS to identify the location of the tones. Figure (4.5) shows the obtained *sparsogram* which is a time-frequency plot that displays only the dominant frequencies in the signal. We get the same sparsogram in both cases, as expected. $S1$-AAFFT samples about 0.94% of the signal whereas CFFS samples about 1.06% of the signal, which is only very slightly larger than $S1$. This experiment demonstrates the efficiency and similarity of the

two methods and supports the proposition made in Section (4.3.3).

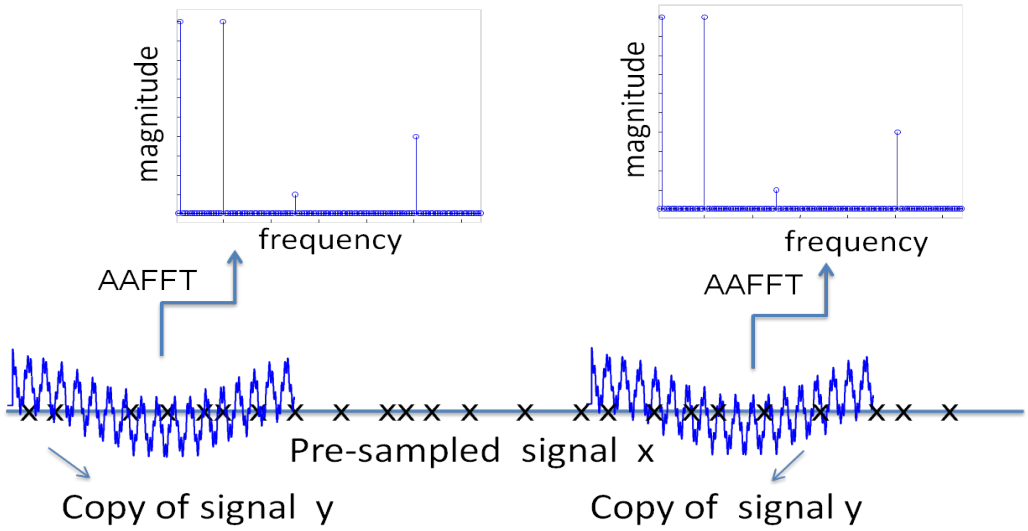

Figure 4.6: Applying CFFS to different blocks of signal x

Arbitrary position of analysis window: While $S1$-AAFFT cannot be applied to compute the dominant tones in any arbitrary block, the CFFS has no such limitation. This is demonstrated in the next experiment as follows. Let y be a signal of length $N = 2^{20}$, with $m = 4$ known dominant frequencies. Note that the specific values of m and N are not integral to the performance of the algorithm. AAFFT has been tested exhaustively and its performance as a function of N,m is completely characterized [24]. Let x be an arbitrary signal of length n with $N \ll n$. Now let $x[n_1, n_2]$ be an arbitrary block of interest of length N. Set $x(n_1 + q) = y(q)$, for $q = 0, 1, \ldots, N-1$. Thus we have placed a copy of the known signal y in the block of interest. The CFFS was then applied and the four dominant frequencies in the block of interest were computed. The obtained values for frequencies and their coefficients match closely with those of the signal y and satisfy the error guarantees of AAFFT. The whole experiment was repeated with different values for n_1 (and corresponding $n_2 = n_1 + N - 1$) and the same results were obtained. Figure (4.6) shows the sketch

of a signal x, pre-sampled in a predetermined manner (according to CFFS), with copies of y placed at arbitrary positions. Applying AAFFT to any block with a copy of y gives the same results thus demonstrating the correctness of CFFS.

Unknown Frequency hopping signal: For simplicity let's assume that the signal has two tones that change at certain intervals which are not known. We are interested in finding the unknown boundaries at which the tones change. In particular, consider two adjacent blocks with f_1 and f_2 as their respective frequencies (see Figure (4.7)). We take an analysis window of length N. The center of the window can be varied and a "binary" search can be performed for the block boundary in the following manner. If the center is to the left of the actual boundary, then the coefficient of f_1 will be higher than that of the f_2. This indicates that the center has to be moved to the right from its current position. This step can be iterated a few times to make the center converge to the actual block boundary.

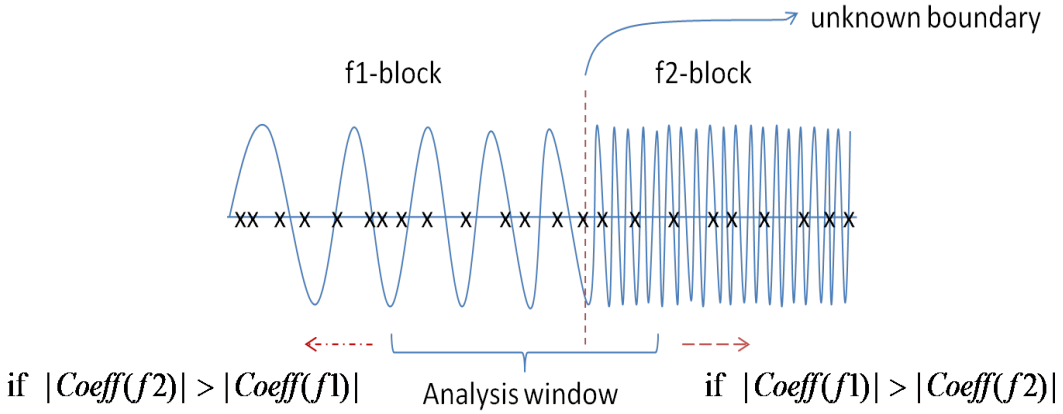

Figure 4.7: Frequency-hopping signal with unknown block boundaries.

Note that looking at the relative ratios is just one way (albeit simple) of doing change point detection in the frequency domain. More sophisticated algorithms ex-

ist ([67],[68],[69]), however they are not designed to work on sub-Nyquist samples and are computationally intensive (with runtime superlinear in signal length). The signal is sampled using the CFFS pattern, thus enabling the application of AAFFT on any analysis window of length N while performing the search. Also the search is not strictly binary since the amount by which $f1$ coefficient is higher than $f2$ can be used to shift the center of the window to the right by an equivalent amount. Once the center converges after a few iterations, we express the error as the distance to the true boundary and determine what percentage of the block this distance is. Table (4.2) displays the error and how the error increases with decreasing SNR.

SNR(dB)	**%Error**
∞	0.39
10	0.58
8	0.70
6	0.78
4	0.79
2	1.56

Table 4.2: Percentage error in boundary identification

Note that even in the case of no noise (infinite SNR) there is some inherent ambiguity in the identification of block boundary. This uncertainty is caused by two factors. First, when the analysis window has portions of both the $f1$-block and $f2$-block, the net signal is no longer sparse due to a sudden change in frequency and has a slowly decaying spectrum. With $m = 2$ the AFFT guarantees that the error made in signal approximation is about as much as the error in optimal 2-term approximation [63]. Hence a slowly decaying spectrum implies more error in the approximation. A second and more important factor is the number of samples actually acquired from the region of uncertainty around the block boundary. From the

entire block, CFFS acquires about 8% samples from the $N = 2^{17}$ present. Assuming these samples are uniformly distributed (which is not true for CFFS), the number of samples present in the region of uncertainty (0.4%) is about 40. In practice, CFFS contains even fewer samples in the uncertainty region (about 30 on average). In terms of samples actually acquired in CFFS, the boundary estimation is off by only a few samples and hence is negligible, as it does not affect the computations. This will be true for any sparse sampling method like CFFS. Furthermore, if the uncertainty were to be reduced to 0.3% say, the boundary identification would improve by only about 6 samples on average, which again is negligible. Hence the boundary identification through the above method is accurate enough for all practical purposes.

4.5 Conclusion and Future Work

We described and proved a sub-linear time sparse Fourier sampling algorithm called the CFFS which along with AAFFT can be applied to compute the frequency content of sparse digital signals at any point of time. Once the block length N is selected, a sub-Nyquist sampling pattern can be pre-determined and the samples can be acquired from the signal (during the runtime if required). The AAFFT can be applied to the samples corresponding to any block of length N of the signal and the dominant frequencies in that block and their coefficients can be computed in sub-linear time. The algorithm requires the block length N to be fixed beforehand. Designing or extending the algorithm to work for different values of N can be considered in the future. Adapting the algorithm to further reduce the computational complexity by using known side information about the signal can also be considered. The algorithm is also highly parallelize-able and can be adapted for hardware

applications. Also, we may be able to extend this sample set generation to the deterministic sampling algorithm described in [12] and the sparse FFT algorithm in [14].

CHAPTER V

Spectrum Sensing Cognitive Radio

5.1 Introduction

In recent years, as a result of numerous emerging wireless applications and services, a scarcity in spectral resources and an increased demand for available spectrum has been witnessed. This scarcity, however, is paradoxical since most of the allocated spectrum remains underutilized at any given time and geographic location. This paradox occurs due to the static nature of the current spectrum licensing scheme which allocates the channels or bands of the spectrum to the primary (licensed) users (e.g. TV broadcast channels, mobile carriers), who do not transmit at all times and locations. This results in spectrum holes or vacancies. These spectral holes are thus free to be used by unlicensed or secondary users. Exploiting this fact, a *Cognitive Radio* (CR) is proposed in [70]. A CR is a "smart" radio which is always aware of its environment and can adapt accordingly. CR systems enable dynamic spectrum access (DSA) and thus improve the overall efficiency of spectrum usage. One of the primary cognitive tasks of a CR is to continuously monitor the frequency spectrum in order to find holes or vacant channels that can be used for secondary transmissions.

A typical solution to spectrum sensing involves filtering the wideband signal with

a bank of narrow-band filters and monitoring each channel using classical techniques such as energy detection (ED) [71] or more recent multi-antenna based detection [72]. However, this approach requires a huge number of RF (radio frequency) components and consumes a large amount of power. An alternative is to sample the entire wide-band signal at Nyquist rate and digitally monitor each channel. A primary challenge with this approach is the requirement of ADCs with very high sampling rates, which can be prohibitively expensive.

Recent advances in compressive sampling (CS) have demonstrated the principle of sub-Nyquist-rate sampling and reliable signal recovery when the signals are sparse or compressible [3]. Since licensed signal transmissions are sparse in the frequency domain, CS techniques can be applied to the cognitive task of spectrum sensing. Exploiting this idea, numerous CS-based CR systems have been proposed in the past five years. Many of them use impractical sampling schemes such as those involving random Gaussian matrices and thus lack an efficient implementation [73, 74, 75, 76, 77]. Some systems use computationally intensive algorithms such as those based on, ℓ_1-norm minimization [73, 75, 76, 78], matrix rank minimization [74], matrix completion [79], PSD (power spectral density) estimation through autocorrelation [73, 76, 80] or Bayesian iterative algorithms [77]. Some methods [74, 75, 79] approximate the wide-band spectrum using a spectrum vector S_f (with length equal to number of channels K) and assume that $S_f = Fx$ where F is a DFT (Discrete Fourier Transform) matrix and x (of length K) is the discretized input wideband signal. This results in a frequency spectrum with very poor resolution and high spectral leakage because of severely time-limiting the input signal. Some CS-CR systems propose cooperative sensing using fusion centers to collect the measurements [79] or signal autocorrela-

tions [76] and perform joint support detection through complicated algorithms.

In contrast to the above, in this work, we develop a spectrum sensing algorithm (Sec. 5.3) that has the following features. The wideband signal is sampled at a sub-Nyquist rate, according to a sampling scheme (Sec. 5.3.1) which can be efficiently implemented using low-rate ADCs. The occupied channels are identified using a simple algorithm (Sec. 5.3.2 and 5.3.3) that processes the signal samples through application of low-dimensional FFTs (Fast Fourier Transforms). The algorithm is easily implemented in a cooperative fashion, with exchange of minimal bits between one-hop neighbors. Also, the algorithm can be implemented in a decentralized fashion without the need for a fusion center. Numerical simulations, in Sec. 5.4, support the theory developed in Sec. 5.3. We conclude with a discussion of future work in Sec. 5.5.

5.2 The Problem Statement

The input wideband signal $x(t)$ is assumed to be band-limited to $[0, F_N]$ where F_N is the maximum frequency in $x(t)$. Note that F_N is also the Nyquist rate. For convenience, we consider only the band of positive frequencies. The developed techniques can be easily applied to a real-valued signal band-limited to $[-F_N/2, F_N/2]$. The wideband spectrum is assumed to be divided into K non-overlapping channels, indexed by $i = 0, 1, .., K-1$. Only $s < K$ of channels are assumed to be occupied, with $I_s \subset \{0, 1, .., K-1\}$ denoting the set containing the indices of the occupied channels. Given F_N, K and s, the problem is to find the set I_s. In practice, s can be assumed to be known approximately from a history of channel occupancy statistics. The problem is depicted in Fig. 5.1, where a spectrum of 120 MHz is

divided among $K = 64$ noisy channels with only $s = 5$ occupied or active channels ($I_s = \{11, 21, 27, 28, 62\}$). The desired output of the spectrum sensing algorithm is plotted on the right, where a 1 indicates channel activity.

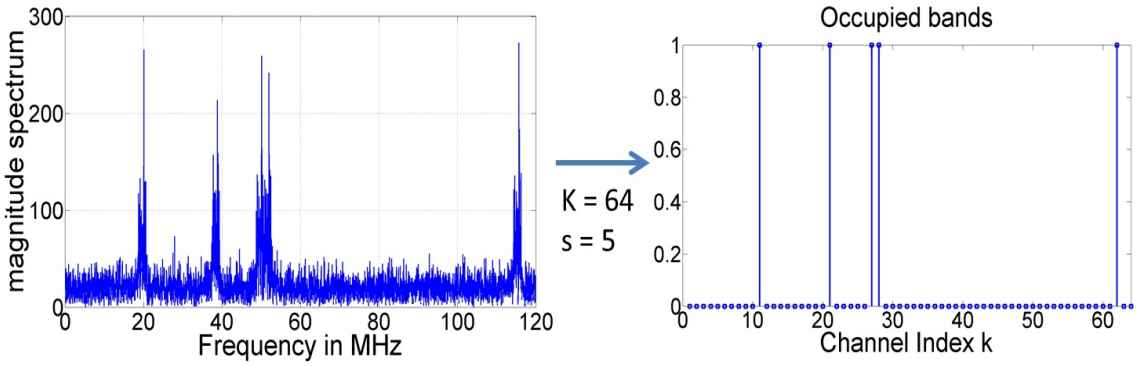

Figure 5.1: (left) The magnitude spectrum of a wideband signal ($F_N = 120$MHz) with $s = 5$ occupied channels in a total of $K = 64$ channels. (right) The desired output of the spectrum detection

If the signal $x(t)$ is observed for a time slot of duration $t_S = N/F_N$, the signal can be discretized as a vector x of length N, with $x[n] = x(n/F_N)$ for $n = 0, 1, .., N-1$. The N-point DFT (Discrete Fourier Transform) of $x[n]$, denoted by $X[f]$ has N frequencies indexed by $f = 0, 1, ..., N-1$. Assuming $N = KR+1$ and ignoring the zero frequency, each channel is made up of $(N-1)/K = R$ discrete frequencies. Since only s channels are occupied, X is sR-sparse[1]. State-of-the-art sub-linear algorithms such as [63] and [14], which reconstruct sparse vector X from random samples of x can be used to identify the occupied bands, however, the required random sampling pattern is challenging to implement in simple hardware. Also, since we only need to detect the s occupied bands and not reconstruct the entire spectrum, we use a similar but much simpler sampling scheme in our algorithm.

[1] A signal is called s-sparse if at most s terms are non-zero

5.3 The Wideband Spectrum Sensing Model

A high-level block diagram of the proposed wideband spectrum sensing scheme is shown in Fig. 5.2, with explicit pseudo-code in Table 5.1. The individual blocks are explained in detail in the sections following. The input signal is sampled according to a structured random sampling pattern (detailed in Section 5.3.1). The samples are processed by R filters ($F_r, r = 0, .., R-1$) whose pass-bands are uniformly interleaved in the frequency domain (see Fig. 5.4). Together, the R filters form, what we call, a uniformly-interleaved filter-bank (UIFB). The structure in the sampling pattern is exploited to perform the filtering operations through low dimensional FFTs (see Section 5.3.2). This process produces random samples of the R outputs of the UIFB, denoted by $x_r[t], r = 0, .., R-1$, where $x_r[t] = (x * F_r)[t]$. As we will see, the UIFB divides the high dimensional input signal into R low dimensional frequency-sparse signals. The s dominant frequencies in these signals are identified by the next block and a K-length output vector b is produced. The indices of the s biggest terms in b give the set of active channels I_s. The robustness of the detection algorithm is improved by taking element-wise median over J independent copies of b. The J copies are produced by the same cognitive radio or by J different cognitive radios when implemented as a collaborative sensing scheme.

5.3.1 The structured random sampling system

The cognitive radio samples the vector $x[n]$ according to the sampling pattern shown in Fig. 5.3 (top portion), where $t_\ell \sim U[0, 1, .., N-1]$ is a uniform random variable for $\ell = 0, 1, .., L-1$, with $L = \mathrm{O}(s \log K)$. For each t_ℓ, the sampling pattern contains an arithmetic progression of size R (under mod-N arithmetic). As we

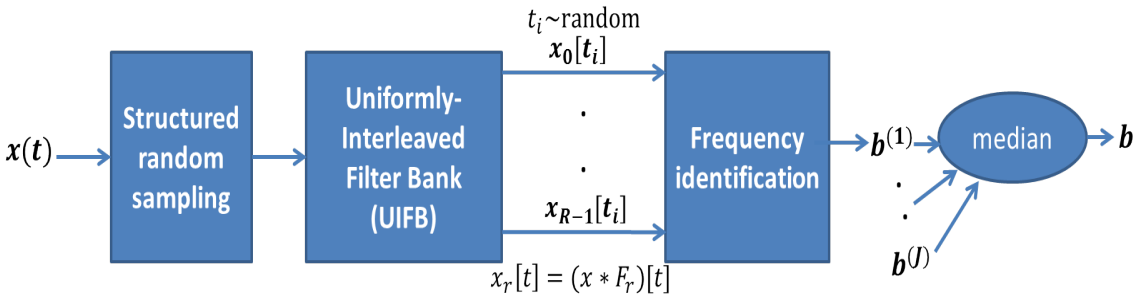

Figure 5.2: Block diagram of the spectrum sensing scheme

will see in Section 5.3.2, the UIFB implementation consists of computing an R-point FFT of each arithmetic progression. The sampling scheme can be easily implemented through a multi-coset system [81], using analog-to-digital converters with a low rate of F_N/K (but with cut-off frequency F_N).

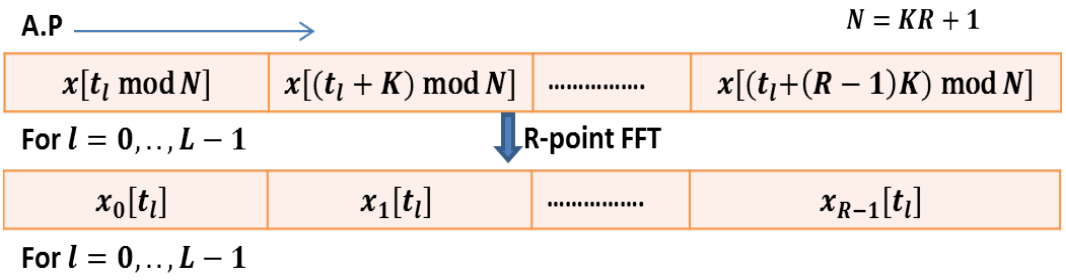

Figure 5.3: (top) Sampling pattern of the proposed structured random sampling scheme and (bottom) random samples of UIFB outputs

5.3.2 The Uniformly-Interleaved Filter bank (UIFB)

An example of an ideal UIFB with $R = 3$ is shown in Fig. 5.4, where it is compared with a regular sub-band decomposition filter-bank with $R = 3$. The UIFB can be conceptually described as a three-step system (see Fig. 5.5). In the first step, the frequencies from different channels in x are uniformly interleaved with each other to give a new signal y. This is illustrated in Fig. 5.6 for $K = 4$ and $R = 6$. This step

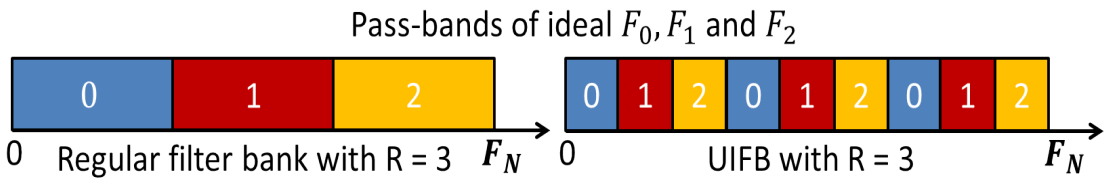

Figure 5.4: Ideal Pass-bands of filters $F_0, F_1, ..$ in a (left) Regular sub-band decomposition filter-bank with $R = 3$ and (right) a uniformly-interleaved filter-bank (UIFB) with $R = 3$.

is carried out through a time dilation $t \mapsto Kt \bmod N$. In the Fourier domain, this translates to the frequency mapping $f \mapsto K^{-1}f \bmod N$, $\forall f = 0, 1, .., N-1$, where $K^{-1} = (K-1)R+1$ is the multiplicative inverse of K under mod-N arithmetic, i.e. $KK^{-1} \bmod N = 1$. The second step consists of passing y through a regular sub-band decomposition filter bank with R band-pass filters that cover the entire spectrum. If h is a low pass filter with R taps whose cutoff frequency is about π/R radians, then the sub-band decomposition filter bank in the second step, can be constructed by modulating h to different frequency bands. That is, $h_r[t] = e^{j(2r+1)\pi t/R}h[t]$ for $r = 0, .., R-1$. For simplicity, we use the boxcar filter with R taps, i.e. $h[i] = 1$ for $i = 0, .., R-1$. It is possible that more sophisticated low-pass filters will sometimes yield better results. In the final step, the frequencies in each filter output are restored to their original places, by carrying out a reverse time dilation $t \mapsto K^{-1}t \bmod N$. The entire process thus has the effect of passing the signal x through a filter bank in which the passbands of different filters are uniformly interleaved.

It is important to emphasize that the algorithm actually implements the three steps of UIFB in a single shot (see Table 5.1). Mathematically, the r^{th} output of the

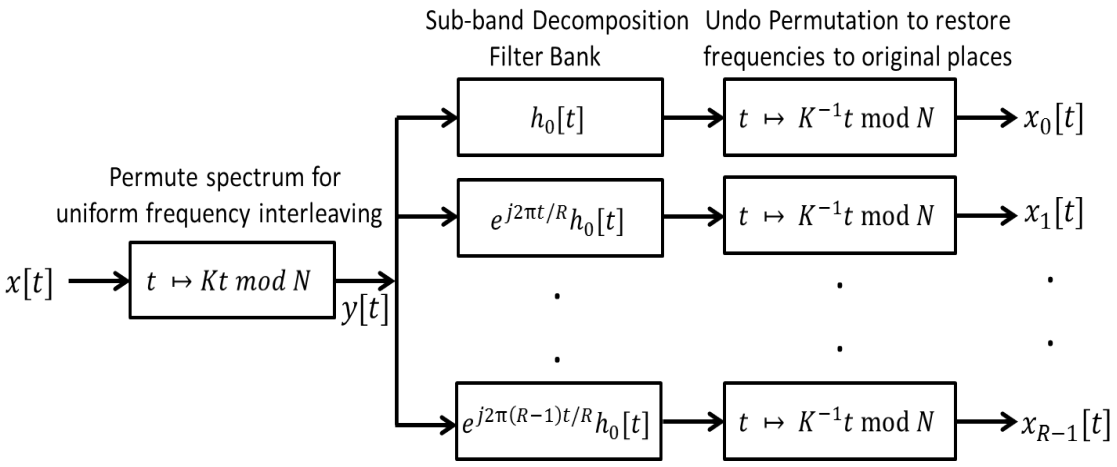

Figure 5.5: A conceptual block diagram of the uniformly-interleaved filter bank

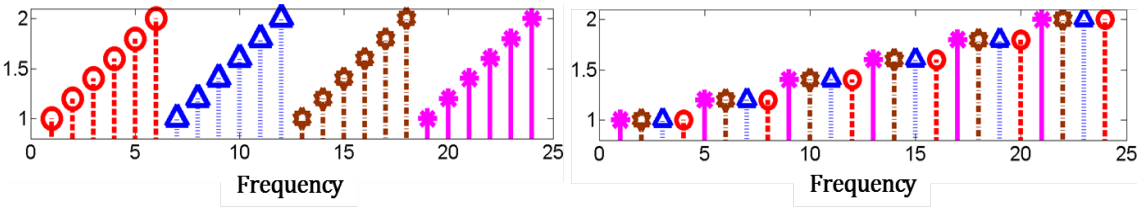

Figure 5.6: (left) Input signal spectrum with $K = 4$ channels ($N = 25$), (right) signal spectrum after uniform frequency interleaving through mapping $f \mapsto 19f \bmod 25$ which corresponds to a time dilation $t \mapsto 4t \bmod 25$

UIFB $x_r[t]$ can be expressed as,

$$x_r[t] = \sum_{i=0}^{R-1} h[i]x[t - Ki]e^{\frac{j(2r+1)\pi i}{R}} = \sum_{i=0}^{R-1} x[t + Ki]e^{-\frac{j\pi i}{R}}e^{-\frac{j2\pi ri}{R}}$$

Given a time point t_ℓ, the outputs of the filter bank $x_r[t_\ell]$ for $r = 0, .., R-1$ can be simultaneously calculated by extracting the signal samples at arithmetic progression as shown in Fig. 5.3, multiplying them by $e^{-j\pi i/R}$, $i = 0, .., R-1$, and computing an R-point FFT. Thus, the R outputs of the UIFB can be randomly sampled at t_ℓ, $\ell = 0, 1, .., L-1$, through computing R-point FFTs on the structured random samples of the input signal.

In Prop. V.1, we prove that the desired uniform interleaving of channel frequencies

can be obtained as shown in Fig. 5.6.

Proposition V.1. *The r^{th} output of UIFB $x_r[t]$ captures the $(r+1)^{th}$ frequency from every channel, for $r = 0, 1, .., R-1$, assuming ideal rectangular filters in the UIFB.*

Proof. For $r = 0, 1, .., R-1$, the r^{th} filter of the UIFB captures $(N-1)/R = K$ frequencies of $x[t]$ that get mapped to the r^{th} filter pass-band, which is made up of the frequencies of $y[t]$ indexed by $\{(rK+i+1), i = 0, 1, .., K-1\}$. Let f_{ri} denote the frequency of $x[t]$ that gets mapped to the frequency $(rK + i + 1)$ of $y[t]$. We have,

$$
\begin{aligned}
f_{ri} &= [(rK + i + 1)K^{-1}] \bmod N \\
&= [(rK + i + 1)((K-1)R + 1)] \bmod N \\
&= [r(K-1)KR + rK + i((K-1)R + 1) + (K-1)R + 1] \bmod N \\
&= [r(K-1)(KR + 1) + r + i(KR + 1) - iR + (K-1)R + 1] \bmod N \\
&= [(r(K-1) + i)(KR + 1) + ((K-1) - i)R + r + 1] \bmod N \\
&= [(r(K-1) + i)N + ((K-1) - i)R + r + 1] \bmod N \text{ (putting } N = KR + 1) \\
&= ((K-1) - i)R + r + 1
\end{aligned}
$$

Thus, the r^{th} filter output $x_r[t]$ captures the frequencies of $x[t]$ indexed by $\{1 + r, 1 + r + R, 1 + r + 2R, ..., 1 + r + (K-1)R\}$, which are the $(r+1)^{th}$ frequencies of all the K channels. For example, the filter output $x_0[t]$ captures the first frequency from all the K channels, given by the indices $\{1, 1 + R, 1 + 2R, ..., 1 + (K-1)R\}$. Thus the UIFB achieves uniformly interleaved filtering.

□

Proposition V.2. *The r^{th} output of UIFB $x_r[t]$ captures the frequencies of $x(t)$ that belong to the class $\{(r+1) \bmod R\}$, for $r = 0, 1, .., R-1$, assuming ideal rectangular*

filters in the UIFB.

Proof. From Prop. V.1, $x_r[t]$ captures the frequencies indexed by $f_{ri} = ((K-1)-i)R + r + 1$ for $i = 0, 1, .., K-1$. Now,

$$f_{ri} \bmod R = (((K-1)-i)R + r + 1) \bmod R = r + 1.$$

Thus f_{ri} belongs to the class $\{(r+1) \bmod R\}$. In other words, $f_{ri} = iR + r + 1$ (from Proposition V.1 with change of variables i to $(K-1)-i$). This is illustrated for $x_0[t]$ in Fig. 5.7.

□

5.3.3 Frequency Identification

From Prop. V.1 and Prop. V.2, we see that each of the K channels contribute a single frequency to each signal $x_r[t]$. Since only s of the K channels are assumed to be occupied, each signal $x_r[t]$ is s-sparse in frequency domain (in practice, due to the non-ideal nature of the filters in the UIFB, each $x_r[t]$ will have non-zero frequencies other than the the dominant s-frequencies captured by the ideal filter). If $b(r)$ denotes the vector containing the coefficients of the K frequencies that are captured by $x_r[t]$, then $b(r)$ is s-sparse. Let $y(r) = [x_r[t_0], ..., x_r[t_{L-1}]]^T$ be the vector that contains the L random samples of $x_r[t]$. We can relate $b(r)$ and $y(r)$ with the underdetermined linear system $B(r)b(r) = y(r)$, where $B(r)_{\ell,i} = e^{j\frac{2\pi}{N} f_{ri} t_\ell}$, $\ell = 0, 1, .., L-1$, $i = 0, 1, .., K-1$ and $f_{ri} = iR + r + 1$ (from Proposition V.2). The linear system $B(r)b(r) = y(r)$ is shown in Fig. 5.8.

The s non-zero terms of each $b(r)$ are identified by applying iterative thresholding (IT) [8] to each set of equations $B(r)b(r) = y(r)$ (see Table 5.1). Since

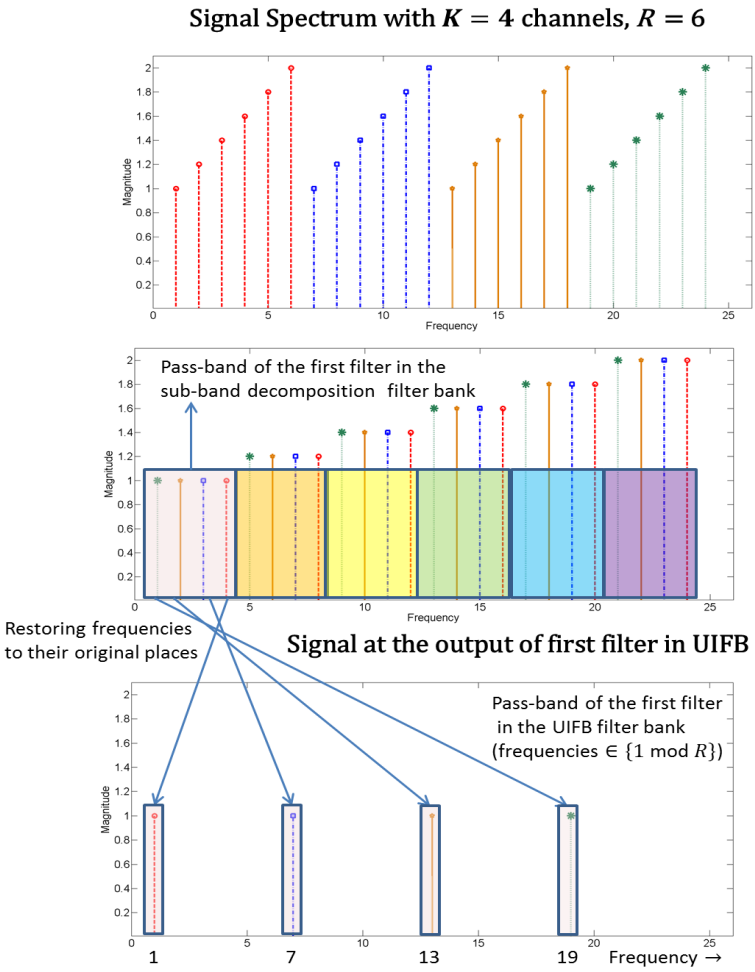

Figure 5.7: (top) Input signal spectrum with $K = 4$ channels ($N = 25$) and $R = 6$ frequencies per channel, (middle) signal spectrum after uniform frequency interleaving through mapping $f \mapsto 19f \bmod 25$ which corresponds to a time dilation $t \mapsto 4t \bmod 25$. Also shown are the $R = 6$ pass-bands of the sub-band decomposition filter bank, (bottom) signal spectrum at the output of the first filter in the UIFB.

$b(r)$,$r = 0, 1, .., R-1$ are jointly sparse, i.e. they have the same non-zero support, the IT can be modified and applied in a joint fashion over $B(r)b(r) = y(r)$, $\forall r$. However for simplicity and to achieve parallelize-ability, we apply IT to obtain an estimate $\tilde{b}(r)$ of each $b(r)$ and combine the outputs at the end utilizing their joint sparsity.

For a vector z, $z_{(s)}$ is defined as the best s-term approximation to z, which can

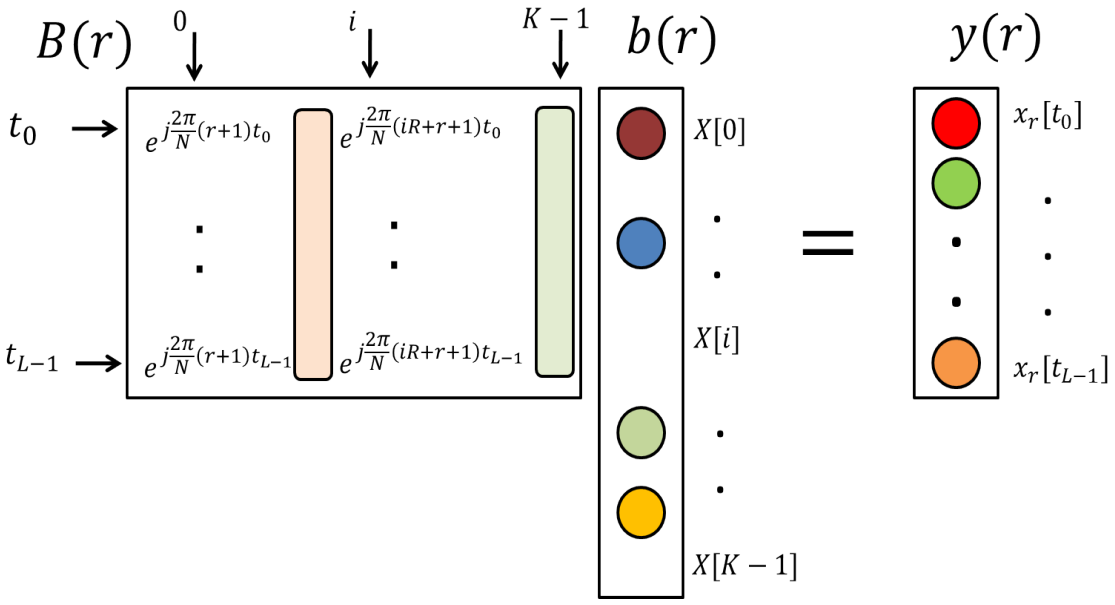

Figure 5.8: Figure showing the various terms in the linear system $B(r)b(r) = y(r)$.

be obtained by setting all the elements of z to zero except the dominant (in magnitude) s-terms. The function $\Phi(a)$ is the indicator function which is defined as 1 if $a \neq 0$ and zero otherwise. The algorithm gets an estimate of the non-zero terms in $b(r)$ by performing the multiplication $B(r)^H y(r)$ and refines this estimate with each iteration. We observed that $reps = 5$ is enough in practice. At the end of iterations, the algorithm converts $\tilde{b}(r)$ into a votes vector by using the indicator function $\Phi(.)$. Hence, the votes vector $\Phi(\tilde{b}(r))$ consists of $1's$ at the indices corresponding to the active channels identified from $\tilde{b}(r)$. Since, $b(r)$ are jointly sparse, we can obtain a more robust estimate of the active channels by performing $b = \sum_{r=0}^{R-1} \tilde{b}(r)$. The set of active channels is obtained as $I_s = \text{supp}(b_{(s)})$, where $\text{supp}(z)$ is defined as the set of indices where z is non-zero.

Proposition V.3. *If $L = O(s \log K)$, then the frequency identification algorithm finds the correct set I_s, with high probability.*

input: $\{t_\ell, \ell = 0, 1, .., L-1\}$, K, R, s.
output: b (length K, sum of votes from each $\tilde{b}(r)$)
UIFB: (from Sec. 5.3.1 and Sec. 5.3.2) for $\ell = 0, 1, .., L-1$, define $z_r = x((t_\ell + rK) \text{mod} N)$, $\forall r = 0, 1, .., R-1$ $[x_0[t_\ell], .., x_{R-1}[t_\ell]] = \text{FFT}\left\{\left[z_r e^{-j\pi r/R}, r = 0, 1, .., R-1\right]\right\}$ **Frequency identification:** (from Sec. 5.3.3) for $r = 0, 1, .., R-1$, $\tilde{b}(r) = \underline{0}$, residual $e(r) = y(r)$ for $i = 1, .., reps$, $\tilde{b}(r) = [B(r)^H e(r) + \tilde{b}(r)]_{(s)}$ $e(r) = y(r) - B(r)\tilde{b}(r)$ $\tilde{b}(r) = \Phi(\tilde{b}(r))$ $b = \sum_{r=0}^{R-1} \tilde{b}(r)$

Table 5.1: The Spectrum Sensing Algorithm

Proof. If $L = \text{O}(\epsilon^{-2} s \log K)$, then the algorithm correctly identifies the s-non zero terms of each $b(r)$ with probability greater than $1 - \text{O}(\epsilon^2)$ (from a similar theorem in [16]). The failure probability $\text{O}(\epsilon^2)$ is further reduced to $\text{O}(\epsilon^2)/R$ as b combines the R different estimates of b (namely $\tilde{b}(r)$,$r = 0, 1, ..., R-1$). □

Each frequency identification block has a run-time of $\text{O}(sK \log K)$. The run-time of the spectrum sensing algorithm is thus dominated by the term $\text{O}(sRK \log K)$. The entire process is illustrated for a signal with $K = 4$ and $s = 2$ in Fig. 5.9 (assuming ideal conditions, i.e. the signal sparsity is $sR = 12$ and each $x_r[t]$ has only $s = 2$ non-zero frequencies).

5.3.4 Improving robustness through median operation

The robustness of the algorithm is improved further by combining J independent copies of vector b. These copies are obtained by an individual cognitive radio which observes the signal for J time slots each of duration $t_S = N/F_N$. This assumes that

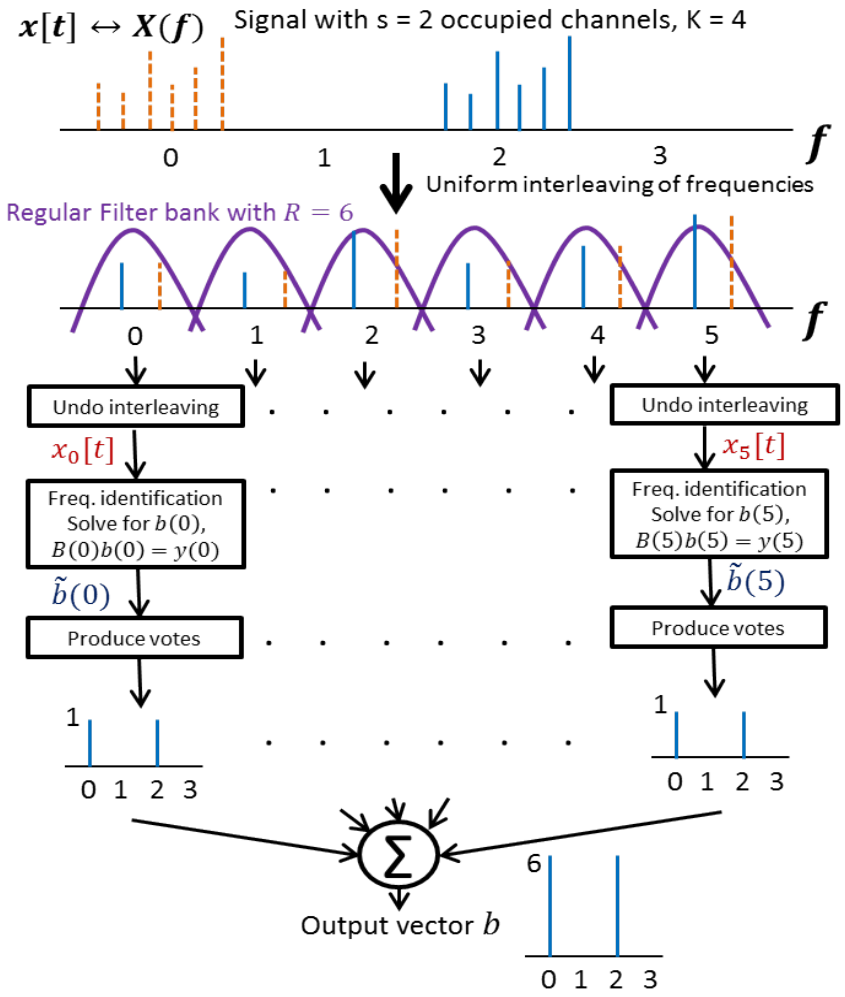

Figure 5.9: The spectrum detection scheme illustrated for a signal with $s = 2$ channels occupied in a total of $K = 4$, for $R = 6$.

the wide band spectrum is either stationary or varying very slowly during the J time slots. When implemented in a collaborative fashion, the J copies of b are obtained by one-hop neighbors (Assuming each CR has J one-hop neighbors). Thus $K\log_2 R$ bits are exchanged between one-hop neighbors. The number of bits that are communicated can be reduced further by employing variable length coding techniques.

If $b^{(i)}$ denotes the i^{th} copy of vector b, $i = 0, 1, .., J-1$, then the final output b is

obtained as

$$b = \text{median}(b^{(0)}, b^{(1)}, .., b^{(J-1)})$$

The active channel set is then obtained as $I_s = \text{supp}(b_{(s)})$.

The advantage of taking a median instead of mean can be demonstrated with the following simple example. Let W be a random variable with unknown mean μ and variance σ^2. Say, we want to estimate W from N independent observations of itself with failure probability of δ. Let $U = \text{mean}(W_1, W_2, .., W_N)$ be the estimate through mean and $V = \text{median}(W_1, W_2, .., W_N)$ be the estimate obtained through taking median. Lets say the estimate fails if it is more than 2 standard deviations away from mean. $\mathbb{P}\text{r}(W_i \text{ bad}) = \mathbb{P}\text{r}(|W - \mu|^2 > 4\sigma^2) \leq 1/4$, by Chebyshev inequality. Now, $\mathbb{P}\text{r}(U \text{ bad}) = \mathbb{P}\text{r}(|U - \mu|^2 > 4\sigma^2) \leq 1/4N$. Equating $\delta = 1/4N \implies N = 1/4\delta$. Lets do the same calculations for V and see how many measurements are needed to get the same failure probability. $\mathbb{P}\text{r}(V \text{ bad}) = \mathbb{P}\text{r}(\text{ more than half of } W_i\text{'s are bad }) \leq \exp(-2N(0.5 - 0.25)^2)$. Equating to δ we get $N = 8\ln(1/\delta)$. For $\delta = 10^{-3}$ for example, U-method needs 250 measurements whereas V-method needs 55 measurements. Taking median has advantage of requiring $O(\ln(1/\delta))$ measurements versus $O(1/\delta)$ required for mean.

Proposition V.4. *If $J = O(log(\frac{1}{\delta}))$, then all the active channels are correctly identified with probability greater than $1 - \delta$.*

Proof. From Proposition V.3, all the active channels are correctly identified in each $b^{(i)}$ with high probability. Let this probability be $p > 0.5$. Then $\mathbb{P}\text{r}$(all active channels are correctly identified in b) = $\mathbb{P}\text{r}$(more than half of $b^{(i)}$ correctly identify all the active

channels) $\geq 1 - e^{-2J(p-0.5)^2}$ (from Chernoff bound). Now, $1 - e^{-2J(p-0.5)^2} \geq 1 - \delta$ for $J = \mathrm{O}(\log(\frac{1}{\delta}))$. □

5.4 Simulation Results and Discussion

The input signal is generated using the following model:

$$x[n] = \sum_{i \in I_s} a_i x_0[n] e^{j2\pi \frac{iF_N}{K} \frac{n}{F_N}} + \xi[n]$$

where $\frac{iF_N}{K}$ correspond to center frequencies of different occupied channels for $i \in I_s$ and $\xi[n]$ is additive white Gaussian noise. $x_0[n]$ is a signal composed of randomly chosen off-grid[2] frequencies $\in [0, F_N/K]$ with random amplitudes and phases. The coefficients a_i correspond to channel gain between the i^{th} primary transmitter and the cognitive radio. Hence, $x[n]$ is modeled, by taking a random multi-tone signal with appropriate frequencies, and translating it in frequency domain to different bands that are to be occupied.

In the following simulations, $F_N = 120$MHz with $K = R = 64$, $s = 5$ and a_i are chosen to be comparable to each other. Note that the values of F_N, etc., are chosen just as an example and are not critical to the performance of the algorithm. The probability of detection[3] P_d and probability of false alarm[4] P_f are used as performance metrics.

[2] An off-grid frequency does not lie on the Nyquist grid of frequencies and causes spectral leakage when input signal is time-limited to a finite window

[3] $P_d = \mathbb{P}r($ all occupied channels are correctly detected $)$

[4] $P_f = \mathbb{P}r($ any of the vacant channels are falsely detected as occupied $)$

5.4.1 Varying Sub-sampling Ratio

In the first experiment, the SNR[5] of each occupied channel is about -2 dB. The probability of detection P_d is calculated empirically over 200 repetitions. We repeat the same experiment with $J = 1, 3, 5$ and 9. The plots for P_d and P_f versus sub-sampling ratio ($LR/N \approx L/K$) are shown in Fig. 5.10. As can be seen from the figure, in the case of $J = 1$, for sub-sampling ratios greater than 0.15, P_d is higher than 0.5 (demonstrating Prop. V.3) and P_d approaches a value close to 1 at $L/K = 0.5$. When J is increased to 3, we see substantial improvement for all sub-sampling ratios that had $P_d > 0.5$ for $J = 1$ (demonstrating Prop. V.4). P_d quickly approaches 1 at $L/K \approx 0.33$. Further improvements in P_d can be obtained by increasing J.

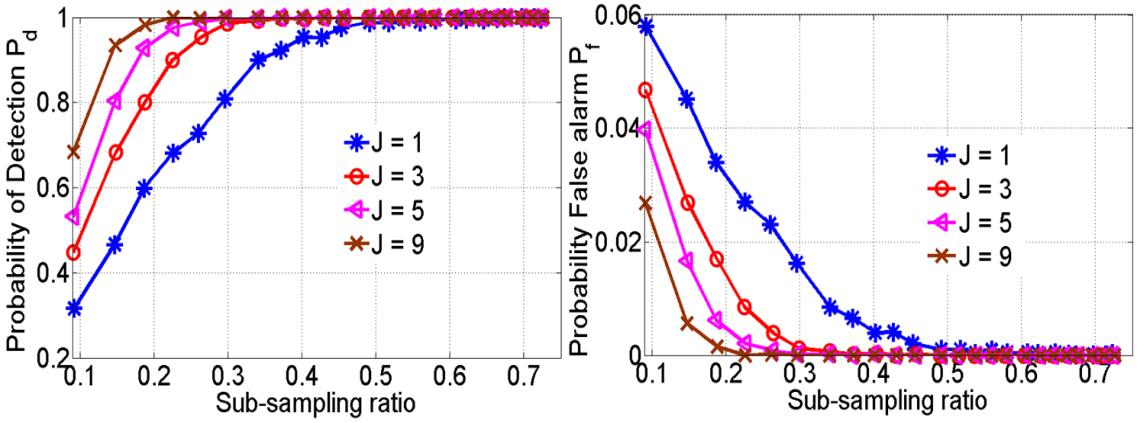

Figure 5.10: P_d (left) and P_f (right) vs. sub-sampling ratio for $J = 1, 3, 5, 9$

5.4.2 Varying Input SNR

In the next experiment, we compare the performance of our scheme (with $L/K = 0.35, 0.3, 0.25$ and $J = 5$) with the Nyquist rate energy detector, for various values

[5]SNR(dB) of each channel is calculated as the ratio of the power in channel and the power of noise $\xi[n]$

of SNR ranging from -25dB to 5dB. In a Nyquist rate energy detector, the received signal is sampled uniformly at Nyquist rate, passed through K narrowband band-pass filters and energy of each filter output is then monitored. A primary user is detected as present, if the output energy of the corresponding filter is among the top s values. The plots are provided in Fig. 5.11. As can be seen from the figures, the performance of the proposed scheme closely follows that of a Nyquist rate energy detector at different SNR values and gives the same performance when SNR > -2dB for $L/K = 0.25$. This improves to SNR > -10dB for $L/K = 0.35$. Note that in the above experiments, for example, $J = 3$ copies can also be viewed as obtained by an individual cognitive radio which observes and samples the signal for 3 time slots each of duration N/F_N.

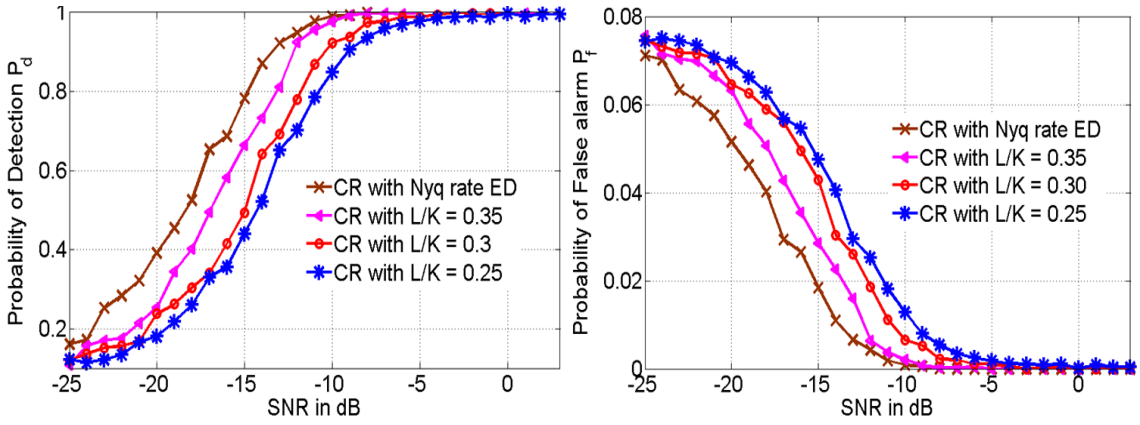

Figure 5.11: P_d(left) and P_f(right) vs. SNR for Nyquist-rate ED and for proposed scheme with $J = 5$, $L/K = 0.35, 0.3, 0.25$

The performance of our scheme is better than those in [80, 78, 74, 75], even though we use a more realistic setting that does not ignore spectrum leakage due to time limitation. It is not clear if we perform better or worse than [73, 76, 79], due to differences in measurement schemes and performance metrics.

5.4.3 Varying R (Number of Frequencies per Channel)

Since the signal $x(t)$ is observed only for a finite window of time duration $t_S = N/F_N$ (with $N = KR + 1$), the discretized signal x exhibits spectral leakage in its DFT. The spectral leakage due to time-limiting is illustrated for a baseband signal (of band width $2W$) in Fig. 5.12.

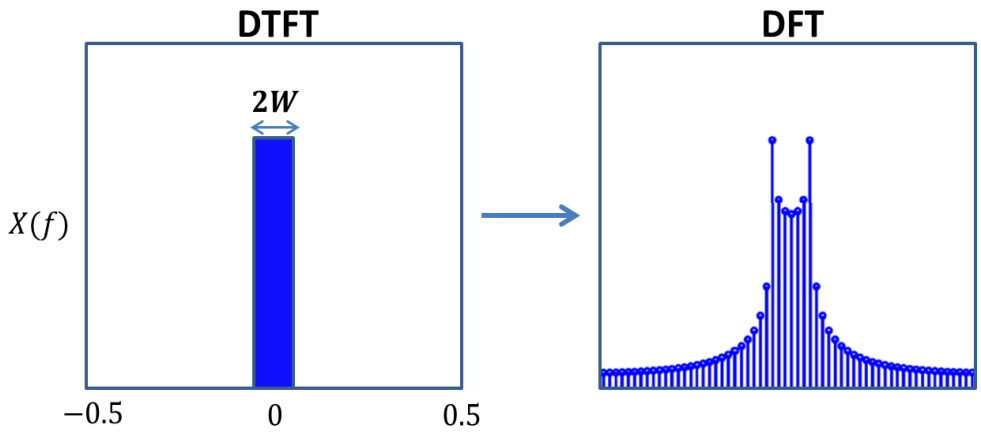

Figure 5.12: (left) DTFT of a bandlimited signal with bandwidth $2W$, (right) DFT of the same signal observed in a limited time window

Quantitatively, the spectral leakage can be calculated as the ratio between the energy that spilled out of the channel and the total energy of the baseband signal. Increasing the value of R, increases N and the window duration t_S, thus decreasing the amount of spectral leakage. Hence the performance of the algorithm improves as R increases. Increasing R, however, also increases (linearly) the runtime of the algorithm. Hence, there is a trade-off involved in the choice of R. In Fig. 5.13, the probability of detection P_d (for $J = 3$) and the spectral leakage (as a fraction of total energy in an occupied channel) are both plotted on y-axis. It can be observed that the probability of detection P_d increases as spectral leakage decreases (with increas-

ing R), and eventually approaches its maximum value of 1 when R is large enough. In accordance with Fig. 5.13, a value of R around 64 seems prudent as there is no advantage in increasing the value further.

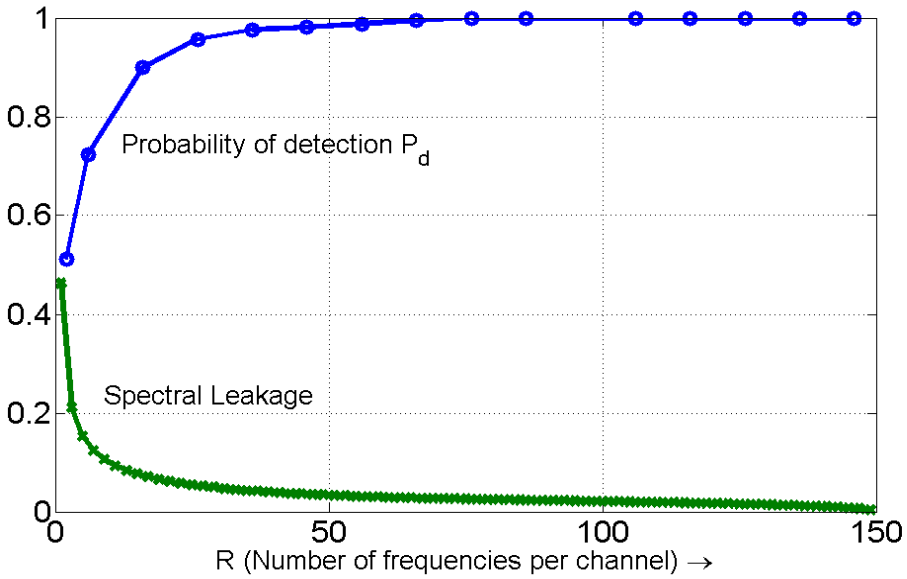

Figure 5.13: Probability of detection P_d versus R and Spectral leakage (expressed as a fraction of total energy in an occupied channel) versus R

5.4.4 Simple Heuristics for Estimating s (Number of Occupied Channels)

The algorithm developed in Sec. 5.3 treats s as an input parameter, assuming its knowledge from a history of channel occupancy statistics. However, this value of s, hereafter referred to as s_{in}, may slightly be in error. Fig. 5.14 shows the performance of the algorithm when s_{in} is in error.

In this section we present simple heuristics to estimate the correct value of s. We assume that $s_{in} > s$.

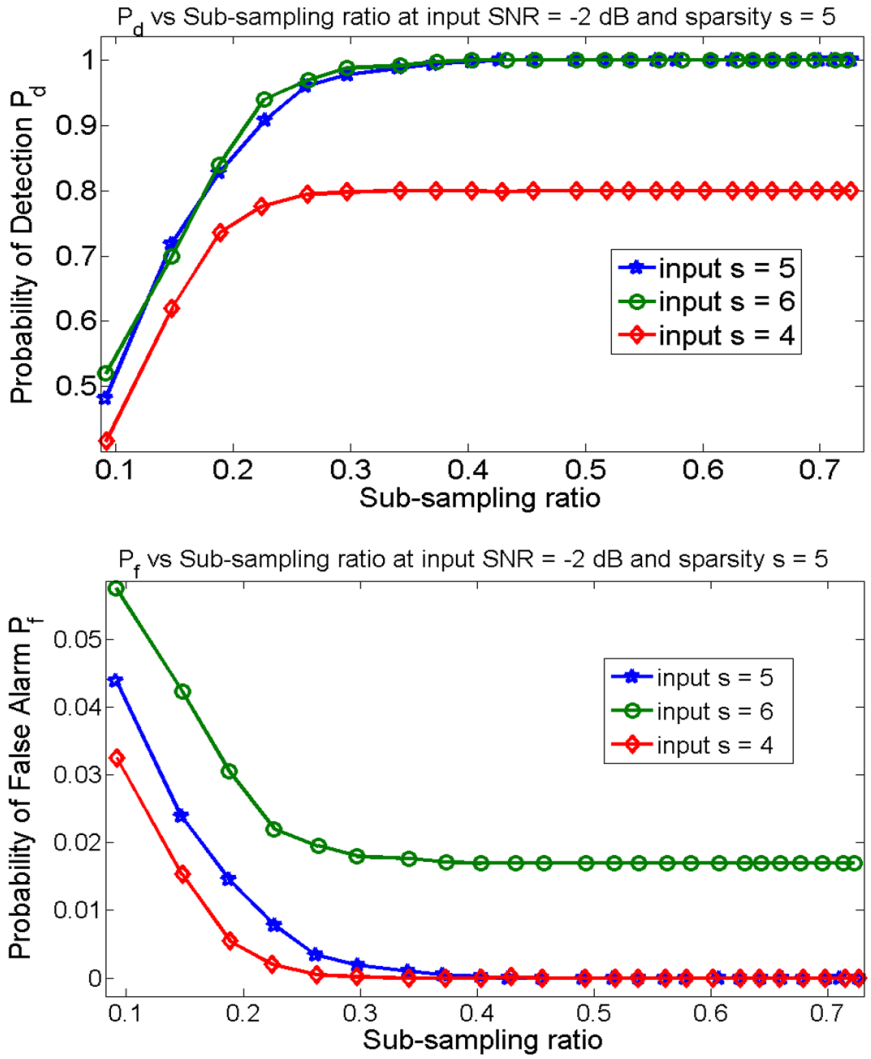

Figure 5.14: P_d(top) and P_f(bottom) vs. Sub-sampling ratio for proposed scheme with $J = 3$, input $SNR = -2$ dB, $s = 5$ and different values of s_{in}.

Heuristic A

Let $\tilde{s}$ denote the estimated value of s. If $b^{(j)}$ denotes the output vector b obtained by CR number j (as in Sec. 5.3.4), for $j = 0, 1, .., J-1$, we calculate a new vector d as follows,

$$d = \text{sum}(b^{(0)}, b^{(1)}, .., b^{(J-1)}).$$

$\tilde{s}$ is then obtained as,

$$\tilde{s} = \text{argmax}_i |d_{[i]} - d_{[i+1]}|$$

where $d_{[i]}$ is the i^{th} largest element of d. In other words, the output votes vectors b (from Table 5.1) for different CRs are added to obtain d. The vector d is sorted in a descending order and the difference between consecutive elements is calculated. If $s = 5$, then the fifth consecutive difference $|d_{[5]} - d_{[6]}|$ is expected to be the largest in magnitude. The active channel set can then be obtained as before in Sec. 5.3.4 using the estimated value $\tilde{s}$ in place of s.

Fig. 5.15 plots the P_d and P_f versus Sub-sampling ratio for the algorithm with and without the estimation of $\tilde{s}$. For reference, P_d and P_f when $s_{in} = s = 5$ are also plotted. As expected, when $s_{in} = 8$ and without estimation, P_d improves while P_f degrades, since the algorithm tries to find $8(> s)$ channels that are occupied. With estimation, both P_d and P_f are brought closer to the case when $s_{in} = s$.

Heuristic B

If $b^{(j)}$ denotes the output vector b obtained by CR number j (as in Sec. 5.3.4), for $j = 0, 1, .., J-1$, we calculate a new vector d as follows,

$$d = \sum_{j=0}^{J-1} \phi\left(b^{(j)}_{(s_{in})}\right)$$

where $z_{(s)}$ is obtained by setting all the elements of z to zero except the dominant (in magnitude) s-terms and $\phi(z) = 1$ if $z > 0$, $\phi(z) = 0$ if $z = 0$. For a chosen threshold $\lambda \in [0, J]$, $\tilde{s}$ is then obtained as,

$$\tilde{s} = \sum_{i=0}^{K-1} \phi(d_i \geq \lambda)$$

where $\phi(\text{true}) = 1$ and $\phi(\text{false}) = 0$.

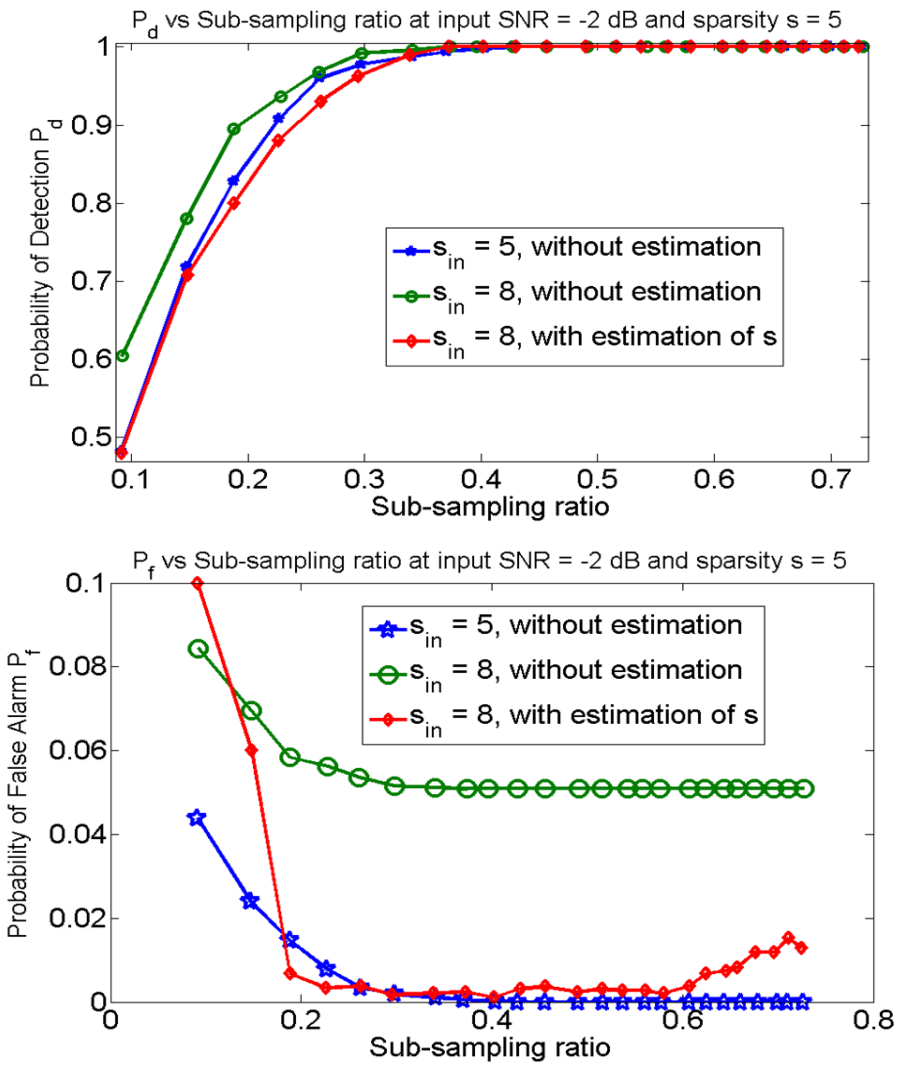

Figure 5.15: P_d(top) and P_f(bottom) vs. Sub-sampling ratio for proposed scheme with $J = 5$, input $SNR = -2$ dB, $s = 5$, $s_{in} = 8$, with and without the estimation of $\tilde{s}$ using Heuristic A.

In other words, in the first step, each cognitive radio j votes for all the s_{in} occupied channels using its output vector $b^{(j)}$. In the second step, $\tilde{s}$ is obtained as the number of channels that received more than λ votes. The net $\tilde{s}$ is chosen to be the maximum among this value and the one estimated by Heuristic A. Once s is estimated, the final output vector is determined by the median operation described in Sec. 5.3.4.

Fig. 5.16 plots the P_d and P_f versus Sub-sampling ratio for the algorithm with and

without the estimation of $\tilde{s}$. The threshold is chosen as $\lambda = \lceil J/2 \rceil$. For reference, P_d and P_f when $s_{in} = s = 5$ are also plotted. Heuristic B performs better in terms of P_f compared to Heuristic A.

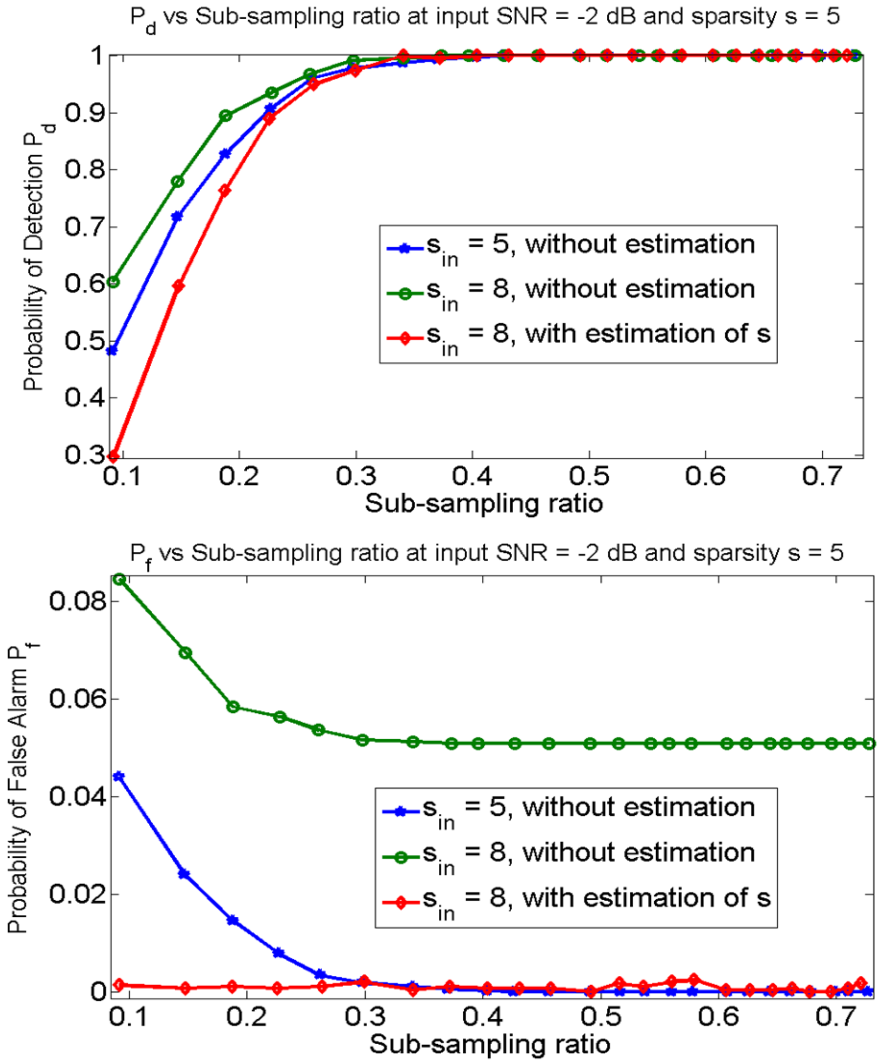

Figure 5.16: P_d(top) and P_f(bottom) vs. Sub-sampling ratio for proposed scheme with $J = 5$, input $SNR = -2$ dB, $s = 5$, $s_{in} = 8$, with and without the estimation of $\tilde{s}$ using Heuristic B.

5.5 Conclusion and Future work

We proposed a novel spectrum sensing scheme for wideband CRs to detect spectrum opportunities, assuming low spectrum utilization. The scheme uses a multi-coset type sampling to efficiently sample the signal at a sub-Nyquist rate close to the channel occupancy. We divided the input signal into several low-dimensional frequency-sparse signals using the newly developed concept of uniformly-interleaved filter bank (UIFB). The UIFB is implemented by "smart" processing of signal samples through low-dimensional FFTs. We solved the frequency identification problem through parallelize-able iterative thresholding techniques. The robustness of the scheme is improved by observing the signal for a longer duration or by collaborating with neighboring one-hop CRs, with minimal pair-wise communication. The algorithm can also be easily extended to estimate the number of occupied channels. Reducing the run-time of frequency identification to a sub-linear time is a future work of interest. Studying the performance under different types of fading channels is another future research direction.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Naraghi, M. Courcy, and M.P. Flynn, "A 9b 14μW 0.06mm^2 PPM ADC in 90nm digital CMOS," *IEEE International SolidState Circuits Conference*, vol. 54, pp. 168–169, 2009.

[2] D.L. Donoho, "Compressed sensing," *IEEE Trans. Info. Theory*, vol. 52(4), pp. 1289–1306, Sep 2006.

[3] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52(2), pp. 489–509, February 2006.

[4] M. Rudelson and R. Vershynin, "Sparse reconstruction by convex relaxation: Fourier and gaussian measurements," *40th Annual Conference on Information Sciences and Systems (CISS)*, 2006.

[5] A. E. Litvak, A. Pajor, M. Rudelson, and N. Tomczak-Jaegermann, "Smallest singular value of random matrices and geometry of random polytopes," *Advances in Mathematics*, vol. 195(2), pp. 491–523, 2005.

[6] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53(12), pp. 4655–4666, December 2007.

[7] D. Needell and J.A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, April 2008.

[8] T. Blumensath and M.E. Davis, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 629–654, September 2008.

[9] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25(2), pp. 21–30, March 2008.

[10] R. A. DeVore, "Deterministic constructions of compressed sensing matrices," *J. Complex.*, vol. 23(4), pp. 918–925, Aug 2007.

[11] A.C. Gilbert, S. Muthukrishnan, and M.J. Strauss, "Improved time bounds for near-optimal sparse fourier representation via sampling," *In Proceedings of SPIE Wavelets XI,San Diego,CA*, 2005.

[12] M. Iwen, "A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods," *In Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

[13] P.K. Yenduri and A.C Gilbert, "Continuous Fast Fourier Sampling," *In Proceedings of Sampling Theory and Applications (SAMPTA), Marseille, France*, 2009.

[14] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse fourier transform," *In Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012.

[15] S. Kunis and H. Rauhut, "Random sampling of sparse trigonometric polynomials, ii. orthogonal matching pursuit versus basis pursuit," *Foundations of Computational Mathematics*, vol. 8(6), pp. 737–763, November 2008.

[16] P.K. Yenduri, A.C. Gilbert, M.P. Flynn, and S. Naraghi, "Rand PPM: A low power compressive sampling analog to digital converter," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5980–5983, May 2011.

[17] P.K. Yenduri, A.C. Gilbert, and J. Zhang, "Model of a sparse encoding neuron," *Twenty First Annual Computational Neuroscience Meeting (CNS)*, Jul. 2012.

[18] P.K. Yenduri, A.C. Gilbert, and J. Zhang, "Integrate-and-fire neuron modeled as a low-rate sparse time-encoding device," *Proceedings of Third International Conference on Intelligent Control and Information Processing (ICICIP)*, Jul. 2012.

[19] P.K. Yenduri and A.C. Gilbert, "Compressive, collaborative spectrum sensing for wideband cognitive radios," *The Ninth International Symposium on Wireless Communication Systems (ISWCS)*, Aug. 2012.

[20] P.K. Yenduri, A. Rocca, A.S. Rao, S. Naraghi, A.C Gilbert, and M.P. Flynn, "A low power compressive sampling time-based analog to digital converter," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Special Issue on Circuits, Systems and Algorithms for Compressive Sensing*, Oct. 2012.

[21] P.K. Yenduri and A.C. Gilbert, "CFFS: A sub-Nyquist sub-linear time sliding window algorithm," *In review, IEEE Signal Processing Letters*, Oct. 2012.

[22] J.A. Michaelsen, J.E. Ramstad, D.T. Wisland, and O. Sorasen, *Low-power Sensor Interfacing and MEMS for Wireless Sensor Networks*, InTech, 1 edition, 2011.

[23] F. Shahrokhi, K. Abdelhalim, D. Serletis, P.L. Carlen, and R. Genov, "The 128-channel fully differential digital integrated neural recording and stimulation interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4(3), pp. 149–161, February 2010.

[24] E. Delagnes, D. Breton, F. Lugiez, and R. Rahmanifard, "A low power multi-channel single ramp ADC with up to 3.2 ghz virtual clock," *IEEE Transactions On Nuclear Science*, vol. 54(5), pp. 1735–1742, October 2007.

[25] T. Fusayasu, "A fast integrating ADC using precise time-to-digital conversion," *IEEE Nuclear Science Symposium Conference Record*, vol. 1, pp. 302–304, 2007.

[26] A.H. Reeves, *Electrical Signaling System*, February 1942.

[27] J. Mark and T. Todd, "A non-uniform sampling approach to data compression," *IEEE transactions on Communications*, vol. 29(1), pp. 24–32, January 1981.

[28] M.Z. Straayer and M.H. Perrott, "A 10-bit 20MHz 38mW 950MHz CT Sigma Delta ADC with a 5-bit noise-shaping VCO-based quantizer and DEM circuit in 0.13μ CMOS," *VLSI Symp. Dig. Tech. Papers*, pp. 246–247, June 2007.

[29] A.A. Lazar and L.T. Toth, "Perfect recovery and sensitivity analysis of time encoded bandlimited signals," *IEEE Transactions on Circuits and Systems-I:Regular Papers*, vol. 51(10), pp. 2060–2073, October 2004.

[30] Aurel A. Lazar and Eftychios A. Pnevmatikakis, "Reconstruction of sensory stimuli encoded with integrate-and-fire neurons with random thresholds," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, 2009, Special Issue on Statistical Signal Processing in Neuroscience.

[31] M. Kurchuk and Y. Tsividis, "Signal-Dependent Variable-Resolution clockless A/D conversion with application to CT-DSP," *IEEE Trans. Circuits and Systems*, vol. 57(5), pp. 982–991, May 2010.

[32] B. Schell and Y. Tsividis, "A continuous-time adc/dsp/dac system with no clock and with activity-dependent power dissipation," *IEEE Journal of Solid-State Circuits*, vol. 43(11), pp. 2742–2481, Nov. 2008.

[33] Y. Li, "A 0.5v signal-specific continuous-time level-crossing adc with charge sharing," *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 381–384, Nov. 2011.

[34] B. Murmann, "A/D converter trends: Power dissipation, scaling and digitally assisted architectures," *IEEE Custom Integrated Circuits Conference (CICC)*, pp. 105–112, 2008.

[35] J.N. Laska, S. Kirolos, M.F. Duarte, T.S. Ragheb, R.G. Baraniuk, and Y. Massoud, "Theory and implementation of an analog-to-information converter using random demodulation," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1959–1962, 2009.

[36] C. Luo and J.H. McClellan, "Compressive sampling with a successive approximation adc architecture," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3920–3923, 2011.

[37] S.R. Becker, "Practical compressed sensing: modern data acquisition and signal processing," in *Dissertation (Ph.D.)*. California Institute of Technology, 2011.

[38] M. Mishali, Y.C. Eldar, and A.J. Elron, "Xampling: Signal acquisition and processing in union of subspaces," *IEEE Transactions on Signal Processing*, vol. 59(10), pp. 4719–4734, October 2011.

[39] F. Chen, A.P. Chandrakasan, and V. Stojanovic, "A signal-agnostic compressed sensing acquisition system for wireless and implantable sensors," *Custom Integrated Circuits Conference (CICC)*, pp. 1–4, 2010.

[40] M. Fornasier and H. Rauhut, *Compressive Sensing*, Springer, 2011.

[41] S.G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 41(12), pp. 3397–3415, 1993.

[42] S. Szczepnski P. Dudek and J.V. Hatfield, "A high-resolution CMOS time-to-digital converter utilizing a vernier delay line," *IEEE Journal on Solid-State Circuits*, vol. 35(2), pp. 240–247, February 2000.

[43] F. Marvasti, *Nonuniform Sampling Theory and Practice*, Kluwer Academic Publisher, 1990.

[44] F. Marvasti, M. Analoui, and M. Gamshadzahi, "Recovery of signals from nonuniform samples using iterative methods," *IEEE Transactions on Signal Processing*, vol. 39(4), pp. 872–878, April 1991.

[45] R.G. Baraniuk and P. Steeghs, "Compressive radar imaging," *IEEE Radar Conf.*, pp. 128–133, Apr. 2007.

[46] F.J. Herrmann, D. Wang, G. Hennenfent, and P. Moghaddam, "Curvelet-based seismic data processing: A multiscale and nonlinear approach," *Geophysics*, vol. 73(1), pp. A1–A5, Feb. 2008.

[47] J. Berent, P. Dragotti, and T. Blu, "Sampling piecewise sinusoidal signals with Finite Rate of Innovation methods," *IEEE Trans. Sig. Proc.*, vol. 58(2), pp. 613–625, February 2010.

[48] A. Bjorck, *Numerical Methods for Least Squares Problems*, Philadelphia: SIAM, 17 edition, 1996.

[49] G. Steidl, "A note on fast Fourier transforms for nonequispaced grids," *Advances in Computational Mathematics*, vol. 9, pp. 337–352, November 1998.

[50] M. Charikar, K. Chen, and M. Farach-colton, "Finding frequent items in data streams," *Theoretical Computer Science*, vol. 312, pp. 3–15, January 2004.

[51] A.A. Lazar and L.T. Toth, "Perfect recovery and sensitivity analysis of time encoded bandlimited signals," *IEEE Trans. on Circuits and Systems-I: Regular Papers*, vol. 51, pp. 2060–2073, 2004.

[52] A.A. Lazar, "Time encoding with an integrate-and-fire neuron with a refractory period," *Neurocomputing*, vol. 58-60, pp. 53–58, Jun. 2004.

[53] A.A. Lazar, E.A. Pnevmatikakis, and Zhou Y., "Encoding natural scenes with neural circuits with random thresholds," *Vision Research Special Issue on Mathematical Models of Visual Coding*, vol. 50(22), pp. 2200–2212, 2010.

[54] E.D. Adrian, *The Basis of Sensation: The Action of the Sense Organs*, Christophers (London), 1928.

[55] D. Attwell and Laughlin S.B., "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow and Metabolism*, vol. 21, pp. 1133–1145, 2001.

[56] V. Balasubramanian and M. J. Berry, "A test of metabolically efficient coding in the retina," *Network: Computation in Neural Systems*, vol. 13(4), pp. 531–552, 2002.

[57] A. Manwani P. N. Steinmetz and C. Koch, "Variability and coding efficiency of noisy neural spike encoders," *BioSystems*, vol. 62(1-3), pp. 87–97, 2001.

[58] G. Gestri, H.A.K. Mastebroek, and W. H. Zaagman, "Stochastic constancy, variability and adaptation of spike generation: performance of a giant neuron in the visual system of the fly," *Biological Cybernetics*, vol. 38(1), pp. 31–40, 1980.

[59] I.F. Akyildiz, W.Y. Lee, M.C. Vuran, and S. Mohanty, "Next generation dynamic spectrum access cognitive radio wireless networks: A survey," *Computer Networks Journal (Elsevier)*, vol. 50, pp. 2127–2159, Sep 2006.

[60] Simon Haykin, *Communication systems*, John Wiley and Sons, 4 edition, 2005.

[61] A.C. Gilbert, M.J. Strauss, J.A. Tropp, and R. Vershynin, "Algorithmic linear dimension reduction in the l1 norm for sparse vectors," *Allerton Conference*, 2006.

[62] G. Cormode and S. Muthukrishnan, "Combinatorial algorithms for compressed sensing," *IEEE Int. Conf. on Information Sciences Systems*, pp. 230–294, April 2006.

[63] A.C. Gilbert, M.J.Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," *IEEE Sig.Proc.Mag.*, vol. 25(2), pp. 57–66, 2008.

[64] M. Iwen, A.C. Gilbert, and M.J. Strauss, "Empirical evaluation of a sub-linear time sparse DFT algorithm," *Commun.Math.Sci*, vol. 5(4), pp. 981–998, 2007.

[65] G.K. Smith and D.M.Hawkins, "Robust frequency estimation using elemental sets," *J.Comput.Graph.Stat*, vol. 9(1), pp. 196–214, 2000.

[66] G. Harikumar and Y. Bresler, "FIR perfect signal reconstruction from multiple convolutions: minimum deconvolver orders," *IEEE Trans.Signal Processing*, vol. 46(1), pp. 215–218, 1998.

[67] O. Mustapha, M. Khalil, G. Hoblos, H. Chafoukand, and D. Lefebvre, "Abrupt change detection algorithm: from theory to implementation," *Colloque Evaluation des performances et matrise desrisques technologiques pour les systmes industriels et nergtiques*, pp. 28–28, May 2009.

[68] S. Rezk, C. Join, S.E. Asmi, M. Dogui, and M.H. Bedoui, "Frequency change-point detection in physiological signals : an algebraic approach," *IJ-STA*, vol. 2(1), pp. 456–468, 2008.

[69] J. Berent, P. Dragotti, and T. Blu, "Sampling piecewise sinusoidal signals with finite rate of innovation methods," *IEEE Trans. Sig. Proc.*, vol. 58(2), pp. 613–625, Feb 2010.

[70] J. Mitola and Jr. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Pers. Commun.*, vol. 6, pp. 13–18, 1999.

[71] R. Tandra and A. Sahai, "SNR walls for signal detection," *IEEE J. Sel. Top. Signal Process.*, vol. 2, pp. 4–17, 2008.

[72] R. Lopez-Valcarce, G. Vazquez-Vilar, and M. Alvarez-Diaz, "Multiantenna detection of multicarrier primary signals exploiting spectral a priori information," *CROWNCOM*, pp. 1–6, Jun 2009.

[73] Y.L. Polo, Ying Wang, A. Pandharipande, and G. Leus, "Compressive wide-band spectrum sensing," *IEEE ICASSP*, pp. 2337 – 2340, 2009.

[74] Y. Wang, Z. Tian, and C. Feng, "Cooperative spectrum sensing based on matrix rank minimization," *IEEE ICASSP*, pp. 3000 – 3003, 2011.

[75] Q. Ling and Z. Tian, "Decentralized support detection of multiple measurement vectors with joint sparsity," *IEEE ICASSP*, pp. 2996 – 2999, May 2011.

[76] Y. Wang, A. Pandharipande, Y.L. Polo, and G. Leus, "Distributed compressive wide-band spectrum sensing," *Info. Th. App. Workshop*, pp. 178 – 183, May 2009.

[77] G. Vazquez-Vilar, R. Lopez-Valcarce, C. Mosquera, and N. Gonzalez-Prelcic, "Wideband spectral estimation from compressed measurements exploiting spectral a priori information in cognitive radio systems," *IEEE ICASSP*, pp. 2958 – 2961, March 2010.

[78] Z. Tian, "Compressed wideband sensing in cooperative cognitive radio networks," *IEEE GLOBECOM*, pp. 1 – 5, Dec 2008.

[79] S. Corroy, A. Bollig, and R. Mathar, "Distributed sensing of a slowly time-varying sparse spectrum using matrix completion," *ISWCS*, pp. 296 – 300, Nov 2011.

[80] M. Rashidi, K. Haghighi, A. Panahi, and M. Viberg, "A NLLS based sub-Nyquist rate spectrum sensing for wideband cognitive radio," *IEEE DySPAN*, pp. 545 – 551, May 2011.

[81] R. Venkataramani and Y. Bresler, "Perfect reconstruction formulas and bounds on aliasing error in sub-Nyquist nonuniform sampling of multiband signals," *IEEE Trans. Info. Th.*, vol. 46(6), pp. 2173–2183, Sep 2000.