Exploring Connections between Sparse Fourier Transform Computation and Decoding of Product Codes

Nagaraj Thenkarai Janakiraman[1], Santosh Emmadi[1], Krishna Narayanan[1] and Kannan Ramchandran[2]
[1] Department of Electrical & Comp. Engg., Texas A&M Univeristy, College Station, TX, U.S.A
[2] Department of EECS, University of California at Berkeley, Berkeley, CA, U.S.A

***Abstract*— We show that the recently proposed Fast Fourier Aliasing-based Sparse Transform (FFAST) algorithm for computing the Discrete Fourier Transform (DFT) [1] of signals with a sparse DFT is equivalent to iterative hard decision decoding of product codes. This connection is used to derive the thresholds for sparse recovery based on a recent analysis by Justensen [2] for computing thresholds for product codes. We first extend Justesen's analysis to d-dimensional product codes and compute thresholds for the FFAST algorithm based on this. Additionally, this connection also allows us to analyze the performance of the FFAST algorithm under a burst sparsity model in addition to the uniformly random sparsity model which was assumed in prior work [1].**

I. Introduction

Let $\underline{x} = (x[0], x[1], \ldots, x[N-1])$ denote a discrete-time signal of length N that is sparse in the Fourier domain with exactly K non-zero Discrete Fourier Transform (DFT) coefficients. Let $\underline{X} = (X[0], X[1], \ldots, X[N-1])$ denote the DFT of $\underline{x}$. Recently, there has been a lot of interest in the design and analysis of computationally-efficient algorithms for computing the location and magnitudes of the K non-zero coefficients from a sub-sampled version of $\underline{x}$ [1], [3], [4].

In this paper, we particularly focus on the Fast Fourier Aliasing-based Sparse Transform (FFAST) algorithm developed by Pawar and Ramchandran in [1], [5]. If a very small probability of error can be tolerated, the FFAST algorithm requires at most $M = O(K)$ samples from $\underline{x}$ and its computational complexity is only $O(K \log K)$ i.e., the sample and computational complexity can both be independent of N. More precisely, it is shown in [1] that a threshold behavior exists, i.e., for $K = O(N^{\delta})$, in the limit of $K, N \to \infty$, recovery with arbitrarily high probability is guaranteed if $M \geq rK$, where r is a constant, which can be interpreted as a threshold.

The key novelty in the FFAST algorithm is the connection that it makes between spectral estimation and peeling-based decoding of low density parity check (LDPC) codes. The FFAST algorithm cleverly uses a Chinese Reminder Theorem (CRT) guided sub-sampling operation to induce aliasing artifacts in the spectrum that look like the parity check constraints of good error-correcting codes like LDPC codes. This enables the use of computationally-efficient decoding algorithms and permits analysis of the the FFAST framework to leverage tools from the design and analysis of codes on graphs. Specifically, it is shown that when the K non-zero coefficients are chosen uniformly at random among the N coefficients, the ensemble of reduced graphs obtained in the FFAST algorithm case is identical to an LDPC code ensemble and, hence, density evolution can be used to obtain the thresholds for the FFAST algorithm.

In this paper, we show that in the very sparse ($K = O(N^{\delta}), 0 < \delta \leq 1/3$) regime, the FFAST algorithm with d stages, i.e., with d sub-sampling patterns (refer to [5] for details) is identical to an iterative decoder for a d-dimensional product code with a single error correcting code in each dimension. This connection allows us to leverage some tools that have been developed for the analysis of product codes to rederive the thresholds for the FFAST algorithm. Particularly, Justesen has analysed the thresholds of iterated product codes in [6] for two-dimensional product codes with arbitrary t-error correcting component codes. We show that extending this analysis to d-dimensional codes results in exactly the same thresholds as in [1], but it provides an alternate characterization of the threshold. This turns out to involve quantities that typically appear in the results on the emergence of k-cores in random graphs. In the less-sparse ($K = O(N^{\delta}),\ 1/3 < \delta \leq 1$) regime, we show that the FFAST algorithm corresponds to an iterative decoder for a carefully defined product code and we show that such a code can also be analyzed and the thresholds derived based from this analysis are identical to those derived in [1].

Interpreting the FFAST algorithm as decoding of a product code also provides insight into the performance of the FFAST algorithm when non-zero coefficients are not randomly chosen, but are bursty such as what may be encountered in many practical applications like spectrum sensing. Here we provide partial results on the recovery of bursty signals when there is exactly one burst or two bursts, i.e., when the spectrum consists of one or two contiguous bands. We provide guaranteed recoverability results for the finite length case and provide thresholds for the 1 and 2 burst cases asymptotically. The performance of the FFAST algorithm for bursty signals appears to be better than those for randomly chosen non-zero coefficients. We show that for $d = 2$ stages when there are two bursts, in the limit of $N \to \infty$, perfect recovery is possible when $K < \alpha\sqrt{N}$ for $\alpha \approx 1$. However, under the random sparsity model for $d = 2$, the probability of recovery cannot be made to be arbitrarily close to 1 for any $\alpha > 0$.

II. System Model

The main idea in the FFAST framework [1] is to exploit the fact that sub sampling in the time domain corresponds to aliasing of the spectrum. If the spectrum is sparse, then the aliased spectrum is also likely to be sparse with very few non-zero coefficients aliasing (overlapping) with one another. By first computing multiple versions of aliased spectra with different under-sampling factors, we can recover the original spectrum if there is a one-to-one mapping between the aliased spectra and the original spectrum. The FFAST algorithm ensures this through the use of the CRT.

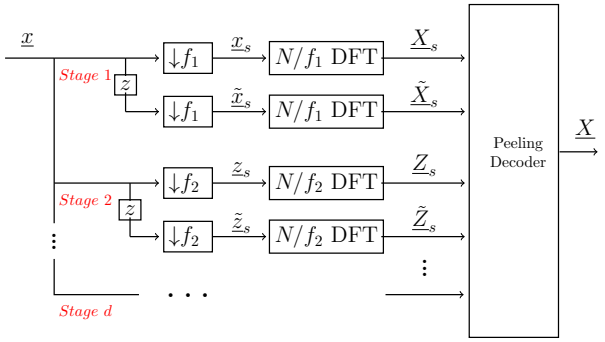

Fig. 1. Block diagram of a FFAST decoder : There are a total of d stages when $N = P_1 P_2 \dots P_d$. $\underline{x}[n]$ and a shifted version of $\underline{x}[n]$ are sub-sampled by f_i in the ith stage and the Fourier transform of the sub-sampled versions (aliased versions of the original spectrum) are computed. A peeling decoder is then used to recover the original DFT coefficients from the aliased spectra.

Let $N = P_1 \times P_2 \times \ldots \times P_d$, where d denotes the number of stages in the FFAST decoder. To keep the discussion simple, we will restrict our attention to the case of $d = 2$ and consider the Block diagram shown in Fig. 1. Instead of using all the samples, in the FFAST algorithm, $\underline{x}$ is first sub-sampled by $f_1 = P_1$ to create a P_2-point signal $\underline{x}_s = (x[0], x[P_1], x[2P_1], \ldots, x[(P_2-1)P_1])$. Let $\underline{X}_s$ denote the P_2-point DFT of $\underline{x}_s$. A second sequence is generated by delaying $\underline{x}$ by t_1 (=1 in the example) and then sub sampling it by P_1 to obtain $\underline{\tilde{x}}_s$ and let its DFT be denoted by $\underline{\tilde{X}}_s$. Similarly, $\underline{x}$ and a shifted version of $\underline{x}$ are also sub-sampled by $f_2 = P_2$ to obtain two other sequences $\underline{z}_s$, $\underline{\tilde{z}}_s$, i.e. we create a total of 4 sequences and their corresponding DFT's as shown in Fig. 1.Notice that the relationship between $X[k]$'s and the 4 aliased DFT coefficients are given by

$$X_s[l_1] = \sum_{i=0}^{P_2-1} X[l_1 + iP_1], \ 0 \le l_1 \le P_2 - 1 \quad (1)$$

$$\tilde{X}_s[l_1] = \sum_{i=0}^{P_2-1} e^{-j\frac{2\pi l_1 (l_1+iP_1)}{N}} X[l_1 + iP_1], \ 0 \le l_1 \le P_2 - 1 \quad (2)$$

$$Z_s[l_2] = \sum_{i=0}^{P_1-1} X[l_2 + iP_2], \ 0 \le l_2 \le P_1 - 1 \quad (3)$$

$$\tilde{Z}_s[l_2] = \sum_{i=0}^{P_1-1} e^{-j\frac{2\pi l_1 (l_2+iP_2)}{N}} X[l_2 + iP_2], \ 0 \le l_2 \le P_1 - 1 \quad (4)$$

The equations above can be visualized using a reduced Tanner graph with the bit nodes as the K non-zero DFT coefficients of $\underline{X}$ and the check-nodes as the observation pairs: $(X_s[l_1], \tilde{X}_s[l_1])$ and $(Z_s[l_2], \tilde{Z}_s[l_2])$. An edge connects a bit node and a check-node if the bit-node coefficient participates in that check node equation. A check-node is called a singleton if there is only one bit-node participating in it. Note that for a singleton check-node, because of the time shift property of DFT, the ratio of an observation pair $\tilde{X}_s[l_1]/X_s[l_1] = e^{-j\frac{2\pi l_1(k_i)}{N}}$, where k_i is the index of the bit-node that is participating in it. The same is true for a singleton check-node with observation pair $(Z_s[l_2], \tilde{Z}_s[l_2])$. Hence, we could determine whether a check-node is a singleton or not from the ratio of the observation pair $\tilde{X}_s[l_1]/X_s[l_1]$ or $\tilde{Z}_s[l_2]/Z_s[l_2]$. If the singletons are detected, we can identify the location of the bit-node from the exponent of the ratio and also find the value of the bit-node and subtract from all the check-nodes that connects to this bit node. This process can be repeated iteratively until all check-node values are zero or all check-nodes are non-singletons. The algorithm gets stuck if all the remaining check nodes are non-singletons. We call the unrecovered positions in the graph as a stopping set.

III. Connections between product codes and the FFAST algorithm

A. Less-Sparse ($K = O(N^\delta), 0 < \delta \le 1/3$) Regime

To maintain clarity, we will first discuss the connection between product codes and the FFAST algorithm for the less-sparse regime for $d = 2$, and then extend to any $d \ge 3$ and the very-sparse regime. Let us consider the K-sparse spectrum X of length $N = P_1 P_2$, where P_1 and P_2 are co-prime. We will first arrange X in $P_1 \times P_2$ matrix given by X'.

Mapping - There exists a mapping from X' to X, $\mathscr{M} : \{0,1,\ldots,P_1-1\} \times \{0,1,\ldots,P_2-1\} \to \{0,1,\ldots,P-1\}$, which relates $X'(i,j)$ with $X(r+1)$, where $r = \mathscr{M}(i,j)$ is given by,

$$\begin{aligned} \mathscr{M}(i,j) &\equiv (j-i)bP_2 + i \pmod N \\ &\equiv (i-j)aP_1 + j \pmod N \end{aligned}$$

where a and b are integers such that $aP_1 + bP_2 = 1$ [7]. The inverse of this mapping from X to X' is given by,

$$(i,j) = \mathscr{M}^{-1}(r) \equiv (r \mod P_2, r \mod P_1).$$

The CRT guarantees that $\mathscr{M}^{-1}$ is indeed a one-to-one and on-to function. The matrix constructed using the above mapping for $N = 4 \times 5$ is shown in Fig. 2.

From Eqn. 1, it can be seen that the l_1th coefficient of $(X_s[l_1], \tilde{X}_s[l_1])$ pair can be used to determine a singleton from the set of values $\{X[l_1 + iP_1], \ 0 \le l_1 \le P_2 - 1\}$. The l_1th column of X' corresponds to a single error correcting code. Similarly, the l_2th row of X' corresponds to a single error correcting code that checks the set of values $\{X[l_2 + jP_2], \ 0 \le l_2 \le P_1 - 1\}$. The Chinese remainder theorem ensures that for every r, the $\mathscr{M}^{-1}$ results in a unique pair (i,j). This essentially means that each symbol $X'(i,j)$ participates in exactly one row and one column or, equivalently, the intersection of every row and every column contains only one symbol. The array X' is what is typically referred to as a product code in the coding literature.

$X[0]$	$X[5]$	$X[10]$	$X[15]$
$X[16]$	$X[1]$	$X[6]$	$X[11]$
$X[12]$	$X[17]$	$X[2]$	$X[7]$
$X[8]$	$X[13]$	$X[18]$	$X[3]$
$X[4]$	$X[9]$	$X[14]$	$X[19]$

Fig. 2. Product code matrix for $d = 2$ and $N = 4 \times 5$

Now that we have established the connection for $d = 2$, let us consider $d \geq 3$ with $N = P_1 P_2 \dots P_d$.

In the less-sparse regime, the down-sampling factors are given by $f_i = P_i,\ 1 \leq i \leq d$. Now, we have d sets of parity check constraints, one from each branch. This could be visualized as a d-dimensional product code given by the following inverse mapping from X to X', namely

$$(i_1, \dots, i_d) = \mathscr{M}^{-1}(r) \equiv (r \mod P_1, \dots, r \mod P_d). \quad (5)$$

Every stage of the FFAST creates $N/f_i = \prod_{j \neq i} P_j$ parity constraints, and hence each parity check constraints is described by a 1-dimensional vector of X'. Again the CRT guarantees that the intersection of the d component codes has exactly one symbol. When $d = 3$, the illustration of a representative parity check constraint in the Y-dimension of the product code matrix is shown in Fig. 3.

B. Very-Sparse ($K = O(N^\delta), 1/3 < \delta \leq 1$) Regime

We first describe the product code formed for $d = 3$, and then extend to any $d \geq 3$. Consider a K-sparse spectrum, $\underline{X}$, of length $N = P_1 P_2 P_3$, and let us arrange $\underline{X}$ in $P_1 \times P_2 \times P_3$ matrix, X' according to the inverse mapping $\mathscr{M}^{-1}$ in Eqn. 5. In the very-sparse regime, the time domain signal, $\underline{x}$, is down-sampled by down-sampling factors $f_1 = P_2 P_3$, $f_2 = P_1 P_3$ and $f_3 = P_1 P_2$. The ith stage in the FFAST generates $N/f_i = P_i$ parity check constraints, and each check contains f_i coefficients of X. Let us examine the following parity check equations generated at the first stage of FFAST to understand how it could be visualized in the product code matrix X'.

$$X_s[l] = \sum_{i=0}^{P_2 P_3 - 1} X[l + iP_1], \quad 0 \leq l \leq P_1 - 1 \quad (6)$$

$$\tilde{X}_s[l] = \sum_{i=0}^{P_2 P_3 - 1} e^{-j\frac{2\pi t_1 (l + iP_1)}{N}} X[l + iP_1], \quad 0 \leq l \leq P_1 - 1 \quad (7)$$

Note that the lth coefficient of the $(X_s[l], \tilde{X}_s[l])$ pair can be used to determine a singleton from the set of values $\{X[l + iP_1],\ 0 \leq i \leq P_2 P_3 - 1\}$ and hence every check is an one-error correcting code by itself. Notice that elements of $X[l]$ in Eqn. 6 lie on a plane in X', where each element on the plane has a remainder l when divided by P_1. Similarly, every check for $d = 3$ in the very-sparse regime can be visualized as a plane in the matrix X' as shown in the Fig. 4. Every coefficient of $\underline{X}$ participates in three checks (planes), one in each branch(dimension), and the intersection of any two planes formed at different stages gives a parity check generated in less-sparse regime.

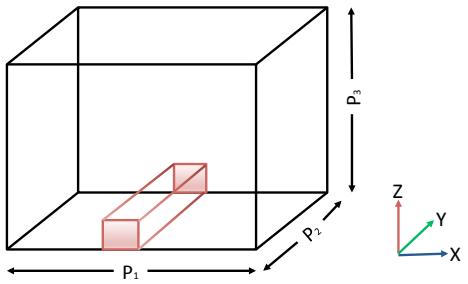

Fig. 3. Illustration of a representative single-error correcting code in dimension Y, with $d = 3$ and operating in less-sparse regime

Now that we have established the connection for $d = 3$, it can be extended to any $d \geq 3$. Consider a K-sparse signal of length $N = P_1 P_2 \dots P_d$ and the down-sampling factors at ith stage of FFAST, $f_i = \prod_{j \neq i} P_j, 1 \leq i \leq d$ and $1 \leq j \leq d$. Let X' be $P_1 \times P_2 \times \dots \times P_d$ matrix whose elements are arranged from X according to $\mathscr{M}^{-1}$. Each parity check equation in this case represents a $(d-1)$-dimensional vector of X' and every coefficient of X participates in d such vectors, one from each stage of FFAST.

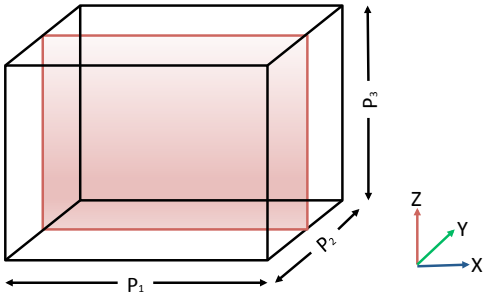

Fig. 4. Illustration of a representative single-error correcting code in dimension Y, with $d = 3$ and operating in very-sparse regime

C. Decoding

It can be seen that the problem of recovering of the K non-zero coefficients is identical to the problem of recovering the errors in a product code assuming the all-zeros codeword is transmitted through a channel that randomly adds K-errors with complex values. It can be seen that an iterative decoding algorithm which decodes all the row codes followed by the column codes in an iterative fashion can be used and this is equivalent to the FFAST algorithm [1]. The iterative process continues until the decoder can not change the codeword(X') anymore from the previous iteration. The decoder stops if all the elements of X' are zero or a non-decodable pattern which contains union of stopping sets (formally defined below) of different sizes is encountered.

Definition 1 (Stopping Set): A stopping set of size t, denoted by $\mathscr{S}_t$, is the set of t points (i, j) in X' that form an non-decodable pattern when an iterative decoder is used for decoding the received product codeword.

Example 2: Following is a stopping set of size 4. Consider $N = 4 \times 5$ and the set $\mathscr{S}_4 = \{(0,0), (0,1), (1,0), (1,1)\}$. Notice that in Fig. 2, with the set of points described in $\mathscr{S}_4$,

we see that the first two rows and columns have two elements each. But each row code is only an one-error correcting code and hence leaving the decoder to get stuck at this point.

IV. Density Evolution and Thresholds

Justesen gave an analysis for iterative decoding of 2-dimensional product codes with t-error correcting RS component codes in [2]. We will use the same approach and extend the analysis to $d-$dimensional product codes in this section. Analytical results for 3-dimensional product codes are compared with that from existing analysis given in [1]. Throughout this section, we will consider a $d-$dimensional product code with individual component code lengths, $P_1, P_2, \ldots, P_d$, all being approximately of same length N^{δ}. To simplify the notation, we will assume that $P_1 = P_2 = \ldots = P$. This is only to simplify the notation and when $N \to \infty$, this approximation does not change the thresholds. The analysis is given for the $d-$dimensional product code in less-sparse regime and the thresholds for the very-sparse regime can be derived in a similar manner.

A. Error pattern as random graph

A $P \times P$ 2-dimensional product code can be described using a bipartite graph with P vertices on left representing P rows and P vertices on right representing P columns and the weighted edges representing the values of the symbols in the corresponding position. As the decoding is independent of codewords and error values, we can work with just the error patterns which translate to random bipartite graphs of $P+P$ vertices where the edges are randomly chosen corresponding to the positions of the random errors.

It is easy to visualize the 2-dimensional product code by its corresponding bipartite graph, but when we extend it to $d-$dimensional ($d \geq 3$) product code, the graph representation is more involved. Each symbol is to be seen as a higher dimensional *hyper* edge connected to d vertices each coming from each of d sets of P^{d-1} vertices. The whole arrangement can be seen as a $d-$partite graph having dP^{d-1} vertices.

B. Decoding

Decoding of $d-$dimensional product code is performed by an iterative decoder, which decodes all the arrays in each dimension in each iteration. There are d component codes corresponding to each of d dimensions in the product code. Each dimension will have P^{d-1} codeword arrays that belong to the same component code. In each iteration of decoding, all of these same component codeword arrays are decoded, before moving on to decoding another component code along another dimension in the next iteration. An array that belongs to a certain component code along some dimension is decoded if it has t or less number of errors in it. Otherwise, it stays unchanged.

C. Analysis

In this sub-section, we will be seeing the threshold analysis of the iterative decoder for the $d-$dimensional product codes. The analysis is extended from the analysis given for 2-dimensional product codes in [2]. The following assumptions are taken to simplify the model for analysis

Assumptions -

1) The errors are randomly distributed in each array. So, when $P \gg t$, the number of errors distributed along each array follows Poisson distribution.
2) In each iteration decoding along a dimension will result in reduction in the number of errors, which is randomly spread across all the arrays of other dimensions. In the equivalent error graph picture, removal of edges from the light vertices in one partition will randomly remove the edges of all vertices of other partitions with same fraction. This approximation becomes exact when P becomes asymptotically large.

Let the number of errors in an array along a dimension be distributed according to a Poisson distribution with parameter m. Then the average number of errors removed when that array is decoded is

$$\sum_{j \leq T} j \mathrm{e}^{-m} m^j / j!$$

The remaining vertices with degree more than t will not be disturbed by decoding and so, will follow a truncated Poisson distribution. The following lemma is adopted from [2] to $d-$dimensional case.

Lemma 3: The degrees of the remaining vertices in a partition will follow a truncated poisson distribution after

1) the decoding along another dimension (partition) removes some randomly chosen edges (assumption 2), and
2) the decoding along the current dimension removes the edges of the newly formed light vertices,

if the degrees of the vertices in that partition follow a truncated poisson distribution initially.

Proof: This proof follows on the similar lines of the proof given in [2]. ■

Definition 4: Let us define $\pi(m)$ as follows

$$\pi(m) = \sum_{j \geq T} \mathrm{e}^{-m} m^j / j!. \tag{8}$$

If the number of errors is distributed according to a Poisson distribution with parameter m before decoding the codes in a dimension, the distribution of errors after the decoding step follows a truncated Poisson distribution with the same parameter m. The mean number of errors after the decoding step can be represented in terms of $\pi(m)$ as follows

$$\sum_{j \geq T+1} j \mathrm{e}^{-m} m^j / j! = m\pi(m).$$

The following Lemma describes the evolution of mean number of errors in the $d-$dimensional product codes and is an extension of the theorem presented in [2].

Lemma 5: If the total number of errors in the received codeword is $W = MP^{d-1}$ initially, the number of errors in each array in each dimension follows a Poisson distribution

with mean M. After first iteration of decoding in one dimension, the number of errors distributed in each array of all other dimensions follows a Poisson distribution with new mean given by,

$$m(1) = M\pi(M).$$

After j iterations of decoding, the number of errors in an array, or the degree distribution, follows truncated Poisson distribution with parameter given by,

$$m(j) = \begin{cases} M\pi(M)\prod_{i=1}^{j-1}\pi(m(j-i)) & \text{if } j < d \\ M\prod_{i=1}^{d-1}\pi(m(j-i)) & \text{if } j \geq d \end{cases} \tag{9}$$

Proof: This proof follows Justesen's proof in [2] and is extended for the $d-$dimensional case. The expected number of errors after first decoding is

$$P^{d-1}\sum_{j\geq T+1} j\mathrm{e}^{-M}M^j/j! = P^{d-1}M\pi(M)$$

using the definition 4. These are randomly distributed along all the arrays (P^{d-1} in number) in each dimension other than the current decoded one. So, the new mean of the number of errors in each array is

$$m(1) = \frac{P^{d-1}M\pi(M)}{P^{d-1}} = M\pi(M).$$

After decoding in stage j, the degree distribution of an array in the current decoded dimension is a truncated Poisson distribution with parameter $m(j-1)$. So, the average number of errors remaining per each array is $m(j-1)\pi(m(j-1))$.

When $j < d$, not all d dimensions are decoded by j^{th} stage, so in every remaining dimension other than the currently decoded, the Poisson parameter of the degree distribution of each array is reduced from its initial value of M to $m(j)$. The reduction factor is equal to the product of reduction factors from the stage 1, i.e.

$$\begin{aligned} \frac{m(j)}{M} &= \prod_{i=1}^{j}(\text{reduction factor in stage } i) \\ &= \prod_{i=1}^{j}\left(\frac{\text{avg. no. of errors in stage } i}{\text{avg. no. of errors in stage } i-1}\right) \\ &= \prod_{i=2}^{j}\left(\frac{m(i-1)\pi(m(i-1))}{m(i-2)\pi(m(i-2))}\right)\left(\frac{M\pi(M)}{M}\right) \\ &= \frac{m(j-1)\pi(m(j-1))}{M} \end{aligned}$$

Assuming the hypothesis is true for all the stages until $j-1$, the following is true by induction.

$$\begin{aligned} m(j) &= M\pi(M)\pi(m(1))\pi(m(2))\ldots\pi(m(j-1)) \\ &= M\pi(M)\prod_{i=1}^{j-1}\pi(m(j-i)). \end{aligned}$$

When $j \geq d$, in every remaining dimension other than the currently decoded, the Poisson parameter of the degree distribution of an array is reduced from $m(j-d)$ to $m(j)$. The reduction factor is equal to the product of reduction factors from the stage $j-d+2$, i.e.

$$\begin{aligned} \frac{m(j)}{m(j-d)} &= \prod_{i=j-d+2}^{j}(\text{reduction factor in stage } i) \\ &= \prod_{i=j-d+2}^{j}\left(\frac{\text{avg. no. of errors in stage } i}{\text{avg. no. of errors in stage } i-1}\right) \\ &= \prod_{i=j-d+2}^{j}\left(\frac{m(i-1)\pi(m(i-1))}{m(i-2)\pi(m(i-2))}\right) \\ &= \frac{m(j-1)\pi(m(j-1))}{m(j-d)\pi(m(j-d))}. \end{aligned}$$

Assuming that the hypothesis in (9) is true for all stages until $j-1$, the following is true by induction.

$$m(j) = \frac{m(j-1)\pi(m(j-1))}{\pi(m(j-d))} \tag{10}$$

$$m(j) = M\prod_{i=1}^{d-1}\pi(m(j-i)). \tag{11}$$

■

Theorem 6: In the limit of large P and fixed t, all $W = P^{d-1}M$ errors are decoded by the iterative decoder when

$$M < \min_m\{m/\pi^{d-1}(m)\} \triangleq c_{d,t}$$

Proof: To show this, we first note that $m(j)$ is a non-increasing sequence since the decoder never adds more errors. Secondly, it can also be seen that $\pi(m)$ is a monotonically increasing function for $m > 0$. This can be verified by computing the derivative of $\pi(m)$ from (8) and noting that $\pi'(m) = \frac{e^{-t}m^l}{l!}$ which is positive for $m > 0$. Thus, $\pi(m(j-i)) \leq \pi(m(j-d)),\ 1 \leq i \leq d-1$. Hence, from (11) we can see that

$$\frac{m(j)}{m(j-d)} = \frac{M\prod_{i=1}^{d-1}\pi(m(j-i))}{m(j-d)} \leq M\frac{\pi^{d-1}(m(j-d))}{m(j-d)}. \tag{12}$$

When $M < \min_m\{m/\pi^{d-1}(m)\}$, the right hand side of the above equation is strictly less than 1 implying that $m(j) < m(j-d)$ and, hence, $m(j) \to 0$ as $j \to \infty$. ■

At the threshold, on the average, there are $c_{d,t}P^{d-1}$ errors and the total number of checks is dP^{d-1}, which means the number of measurements is $2dP^{d-1}$. Thus, the thresholds in terms of the ratio of the number of measurements to the sparsity that can be recovered is given by $\frac{2dtP^{d-1}}{c_{d,t}P^{d-1}} = \frac{2dt}{c_{d,t}}$.

D. Decoding in very-sparse regime

We have already seen the interpretation of constructing $K-$sparse signal in very-sparse regime as decoding of a $d-$dimensional product code with codewords being $(d-1)-$dimensional codeword planes. It is different from the conventional $d-$dimensional product codes as the latter have codewords as $1-$dimensional codeword arrays in $d-$dimensions. However, the analysis of decoding is similar in both cases as the process of iterative decoding essentially same for both the cases.

1) Error graph: The $d-$dimensional product code with $(d-1)-$dimensional codeword planes has P component codes in each of d dimensions, with individual component code length of P^{d-1}. Each symbol participates in d codeword planes. Since the decoding is independent of codewords and error values, we can work with just the error patterns and their error graphs. A corresponding error graph will have hyper edges, chosen according to the random errors in the error pattern, connecting to the vertices, which represent each codeword plane, coming from d sets of P vertices each. This arrangement is a $d-$partite graph with dP vertices.

2) Evolution of the Poisson Parameter: Decoding of the $d-$dimensional product code with $(d-1)-$dimensional codeword planes is done by iteratively decoding all planes in each dimension in an iteration before moving on to decoding in another dimension in another iteration. This is similar to decoding of the conventional product code with $1-$dimensional codeword arrays, so a similar analysis can be conducted here.

The main idea will be to model the number of errors in each codeword plane as having a Poisson (or, truncated Poisson) before (or, after) decoding each codeword and to track the evolution of the parameter of this Poisson distribution. Similar to the less sparse regime, here we can assume that when one of the codeword planes is corrected, it uniformly affects the errors in the codeword planes in other dimensions. Hence, the asymptotic analysis will be identical to that for the less sparse regime. The following results, similar to those in Theorem 5 and Theorem 6 can be shown.

Lemma 7: If the total number of errors in the received codeword is $W = MP$ initially, then the degree distribution, or the number of errors distributed in each $(d-1)-$dimensional plane is according to the truncated Poisson distribution of parameter given by equation 9

Theorem 8: In the limit of large P and fixed t, all $W = MP$ errors are decoded by the iterative decoder when

$$M < \min_{m}\{m/\pi^{d-1}(m)\} \triangleq c_{d,t}$$

At the threshold, on the average, there are $c_{d,t}P$ errors and the total number of checks is dP, which means the number of measurements is $2dP$. Thus, the thresholds in terms of the ratio of the number of measurements to the sparsity that can be recovered is given by $\frac{2dtP}{c_{d,t}P} = \frac{2dt}{c_{d,t}}$. When we calculate $c_{d,t}$ and the corresponding thresholds, $\frac{2dt}{c_{d,t}}$, for $3 \leq d \leq 8$ and $t = 1$, we obtain the values given in Table. I, and we observe that they are very close to the thresholds given by $2d\eta$ in Table IV in [1].

TABLE I

THRESHOLD VALUES

d	3	4	5	6	7	8
$c_{d,1}$	2.455	3.090	3.509	3.823	4.072	4.280
$\frac{2d}{c_{d,t}}$	2.4440	2.5891	2.8498	3.1389	3.4381	3.7383

V. BURSTY SIGNALS

In this section, we describe a procedure to analyze the performance of FFAST when the input signals are bursty in nature, and also establish some bounds for $d = 2$. We will first define some notations required to describe the analysis procedure for signals with two bursts. Note that the term *bursty signals* through out this section refers to signals that have non-zero Fourier coefficients occurring in bursts and it should not be confused with the time domain bursts. Let b be the number of bursts.

A. b=1

Theorem 9: For any finite N and $d = 2$, $b = 1$, any burst of length $K \leq P_1 + P_2 - 1$ is guaranteed to be recoverable

Proof: Without loss of generality, we could assume that the burst starts at (0,0) in X' because of the cyclic property of DFT, and also assume $P_2 < P_1$. The FFAST decoder fails if there exists a sub-matrix of X' which has all the rows and columns filled with more than two coefficients each. In the single burst case, the minimum l for which this occurs is when the burst wraparounds in such a way that the $P_2 \times P_2$ sub-matrix starting at $(0,0)$ is filled with two elements in each row and column as shown in Fig. 5. The length of the burst is sum of lengths of the paths$(1,2,3,4)$ shown Fig. 5. Note that the paths 1 and 2 cover all the P_1 columns once and hence is of length P_1, and the paths 3 and 4 cover all the rows once and hence is of length P_2. So the sum of the path lengths is $P_1 + P_2$. Note that if l is one less than $P_1 + P_2$, there exists a column that has one element in it that is recoverable and the decoder recovers all other coefficients in the subsequent iterations. ■

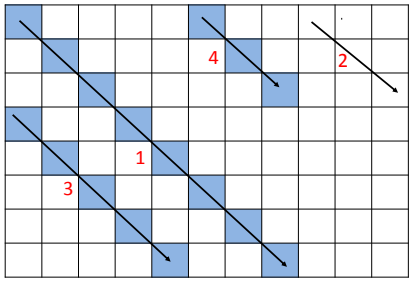

Fig. 5. Illustration of 1-burst positions in X' for the smallest K for which FFAST fails. The blue colored boxes indicate the positions participating in the stopping set.

B. b=2

Let $N = P_1 \times P_2$, and the set of non-zero coefficients that belong to the two bursts be denoted by $\mathscr{B}_1$ and $\mathscr{B}_2$, of lengths k_1 and k_2 respectively. Let the number of symbols (or space) between these two bursts be denoted by s.

We consider the sparse signal $\underline{X}$ in product code structure X', as given by the mapping $\mathscr{M}$ described in Section III. Let us first consider the two bursts $\mathscr{B}_1$ and $\mathscr{B}_2$ such that $\mathscr{B}_1$ starts at the first coefficient of $\underline{X}$, which corresponds to $(0,0)$ in the array X'. A stopping set of size $2n$ contains $2n$ points divided equally in two bursts $\mathscr{B}_1$ and $\mathscr{B}_2$. Consider the case $k_1 < min(P_1, P_2)$, where $\mathscr{B}_1$ has all the points on the diagonal of X' continuously from (0,0) to $(k_1 - 1, k_1 - 1)$. Let us now

solve for the points on $\mathscr{B}_2$ that form part of the stopping set of size $2n$.

Definition 10: The smallest value of k_1 or k_2 such that a stopping set of size $2n$ occurs in received codeword for some value(s) of space, s, between $\mathscr{B}_1$ and $\mathscr{B}_2$, is defined and denoted as K_n. So, for $k_1, k_2 < K_n$, an $\mathscr{S}_{2n}$ never occurs in the received codeword X with two bursts $\mathscr{B}_1$ and $\mathscr{B}_2$.

Let us now consider a stopping set of size $2n$ denoted as $\mathscr{S}_{2n}$. The points in $\mathscr{B}_1$ would be $(a_1,a_1),(a_2,a_2),\ldots,(a_n,a_n)$. Since $\mathscr{B}_1$ and $\mathscr{B}_2$ form a stopping set, without loss of generality, the corresponding points in $\mathscr{B}_2$ must be n ordered pairs whose first indices are fixed to be $\{a_1,a_2,\ldots,a_n\}$. So, the second indices of points in $\mathscr{B}_2$ will be a permutation, $(b_1,b_2,\ldots,b_n) = \phi_j(a_1,a_2,\ldots,a_n)$ $\forall j < n!$ such that $b_i \neq a_i$ $\forall i \in \{1,2,\ldots,n\}$. So, the points in $\mathscr{B}_2$ are $(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)$, let us say, occur sequentially, i.e, $\mathscr{M}(a_1,b_1),\mathscr{M}(a_2,b_2),\ldots,\mathscr{M}(a_n,b_n)$ are points in ascending order in mapped $\mathscr{B}_2$ that belongs to X.

Every stopping set of size $2n$ corresponds to a permutation $\phi_j, j < n!$, and the points in $\mathscr{B}_2$ satisfy the following linear equations corresponding to that permutation, as their maps, $\mathscr{M}(a_i,b_i)$ for $i = 1,2,\ldots,n$, are in increasing order of their position in X.

$$\begin{aligned} b_2 - b_1 \pmod{P_1} &\equiv a_2 - a_1 \pmod{P_2} \\ b_3 - b_2 \pmod{P_1} &\equiv a_3 - a_2 \pmod{P_2} \\ &\vdots \\ b_n - b_{n-1} \pmod{P_1} &\equiv a_n - a_{n-1} \pmod{P_2} \end{aligned} \tag{13}$$

The set of equations(Eqn. refeqn:congruence) for a permutation ϕ_j is denoted by $\mathscr{P}(\phi_j)$. Each equation is an equation in congruence, is translated to a linear equation in the following sense. As we are finding the solution for $\{a_1,a_2,\ldots,a_n\} < P_2$, each congruence equation $x \pmod{P_1} \equiv y \pmod{P_2}$ is equivalent to

$$\begin{cases} x = y & \text{if } x,y \geq 0 \\ P_1 + x = y & \text{if } x < 0, y \geq 0 \\ x = y + P_2 & \text{if } x \geq 0, y < 0 \\ P_1 + x = y + P_2 & \text{if } x,y < 0 \end{cases} \tag{14}$$

Translation of a congruence equation to a linear equation requires adding of P_1 or P_2 if there is a wraparound. A wraparound occurs when an element of the burst that is not at the end of the burst lies on the boundary of the product code matrix X'. It could be seen from Fig. 5 that a wraparound occurs whenever a transition is made from the last element of each path to the first element of the next path. In Eqn. 14, we will use the first equation if there is no wraparounds, second if there is a column wraparound, and the third equation if a row wraparound occurs.

Theorem 11: An $\mathscr{S}_{2n}$ occurs first time in X' for $k_1 = k_2 = K_n$ when k_1 and k_2 are increased, individually, from 1 to P_2 (for $P_2 < P_1$), if there exists a solution for $\mathscr{P}(\phi_j)$ in $\{a_1,a_2,\ldots,a_n\} < P_2$ for some permutation ϕ_j.

Proof: If there exists a solution in $\{a_1,a_2,\ldots,a_n\} < P_2$ for some permutation ϕ_j, then

$$\begin{aligned} K_n &= \min_{\forall \mathscr{P}(\phi_j)} \mathscr{M}(a_n,b_n) - \mathscr{M}(a_1,b_1) \\ &\equiv \min_{\forall \mathscr{P}(\phi_j)} a_n - a_1 \pmod{P_2} \\ &\equiv \min_{\forall \mathscr{P}(\phi_j)} b_n - b_1 \pmod{P_1} \end{aligned}$$

When an $\mathscr{S}_{2n}$ occurs in X', $\mathscr{B}_2$ contains (a_1,b_1) and (a_n,b_n). So we have $k_2 \geq K_n$. Similarly, we have $k_1 \geq K_n$. Hence, an $\mathscr{S}_{2n}$ occurs first time for $k_1 = k_2 = K_n$ when k_1 and k_2 are increased, individually, from 1 to P_2. In that case, the end points of both bursts $\mathscr{B}_1$ and $\mathscr{B}_2$ are in $\mathscr{S}_{2n}$. ■

Corollary 12: An $\mathscr{S}_{2n}$ occurs in X' for $k_1, k_2 \geq K_n$ if there exists a permutation ϕ_j such that $\mathscr{P}(\phi_j)$ have solutions for $(a_1,a_2,\ldots,a_n)$ for some K_n.

Proof: This is easy to see from the above proof, as when $k_1, k_2 \geq K_n$, $\mathscr{B}_1$ and $\mathscr{B}_2$ can include $\mathscr{S}_{2n}$ for a certain permutation ϕ_j. ■

Note: In the above, we observe that the system of equations $\mathscr{P}(\phi_j)$ has n variables, $\{a_1,a_2,\ldots,a_n\}$, in $n-1$ equations. We have one degree of freedom. We chose that to be the starting point of $\mathscr{S}_{2n}$ in $\mathscr{B}_1$. So, we can express all the points that form $\mathscr{S}_{2n}$ with respect to this point. When $k_1 = k_2 = K_n$, this starting point is the starting point of $\mathscr{B}_1$ as in this case the end points of $\mathscr{B}_1$ and $\mathscr{B}_2$ are in $\mathscr{S}_{2n}$ from Theorem 11.

Lemma 13: For any finite $N = P_1P_2$ $(P_1 > P_2)$, $d = 2$ and $b = 2$, there exist no stopping set of size $2n$ for those s values that create only one wraparound in bursts, if the length of the bursts is $l \leq \frac{(n-1)}{n} P_2$, where n is a factor of P_2.

Proof: Consider a stopping set of size $2n$ ($\mathscr{S}_{2n}$) formed by the bursts $\mathscr{B}_1$ and $\mathscr{B}_2$ with $\mathscr{B}_1$ starting at $(0,0)$. Let us assume that only one wraparound happens in $\mathscr{B}_2$. Consider a permutation ϕ_j with $(b_1,b_2,\ldots,b_n) = (a_n,a_1,a_2\ldots,a_{n-1})$, a cyclic shift of the indices in $\mathscr{B}_1$. According to Theorem 11, if there exists a solution for the equations $\mathscr{P}(\phi_j)$, then there exists a stopping set of size $2n$ in X'. Without loss of generality, set $a_1 = 0$. The set of linear equations corresponding to the chosen ϕ_j is given by

$$\begin{aligned} a_2 - 0 &= 0 - a_n + P_2 \\ a_3 - a_2 &= a_2 - 0 \\ a_4 - a_3 &= a_3 - a_2 \\ a_5 - a_4 &= a_4 - a_3 \\ &\vdots \\ a_k - a_{k-1} &= a_{k-1} - a_{k-2} \\ &\vdots \\ a_n - a_{n-1} &= a_{n-1} - a_{n-2} \end{aligned}$$

Notice in the first equation, when the congruence is translated to a linear equation form, there is a P_2 term added to the right hand side, and this is to compensate for the one

wraparound assumed. On solving the set of linear equations we get $a_k = \frac{k-1}{n}P_2, 2 \leq k \leq n$. Hence the minimum burst length for which there is a stopping set of size $2n$ is $a_n = \frac{n-1}{n}P_2$. Also notice that for a_k's to be valid points in X', they must be integers, and hence, n must be a factor of P_2. ■

Example 14: An illustration for the above lemma for $n = 2$ is shown in Fig. 6. The colored boxes indicate the position of the stopping set elements.

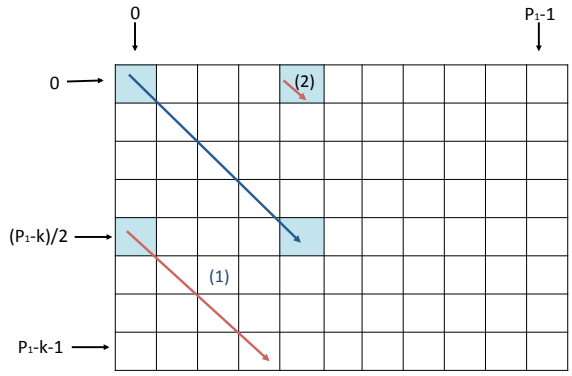

Fig. 6. Illustration of 2-burst positions in X' for the smallest K, for which FFAST fails, particularly when $N = P_1 \times P_1 - l$, where l is an odd number.

Theorem 15: For any finite $N = P_1P_2$ $(P_1 > P_2)$, $d = 2$ and $b = 2$, all bursts of length $l \leq P_2/2$ are recoverable.

Proof: Considering that only one wraparound is allowed, Lemma 13 states that there is no stopping sets for bursts of length $l \leq \frac{n-1}{n}P_2$. Note that $\frac{n-1}{n} \geq 1/2, \forall n \geq 2$. Hence all separations,$s$, corresponding to one wraparounds can be recovered for $l \leq P_2/2$. Also if $l \leq P_2/2$, all the bursts with two wraparounds can always be recovered, as all the rows and columns other than in the $P_2/2 \times P_2/2$ matrix starting at $(0,0)$ has at most one coefficient, and also the elements in the $P_2/2 \times P_2/2$ matrix has two bursts that do not wraparound. Hence there are no stopping sets for $l \leq P_2/2$ in both one and two wraparound conditions, leading to the condition where all bursts of length $l \leq P_2/2$ are recoverable. ■

This result essentially means that even for $d = 2$, there exists a non-trivial threshold in the bursty case as opposed to the random case.

Even though the guaranteed error correction for two bursts is only $l \leq P_2/2$, the fraction of burst with $P_2/2 \leq l \leq P_2$ that are not decodable appears to be small. Therefore, we make the following conjecture.

Conjecture: As $N \to \infty$, for $d = 2$, $b = 2$, the fraction of bursts of length $K \leq P_1 + P_2 - 1$ that are not recoverable vanishes as $N \to \infty$.

VI. Simulation Results

Two sets of simulations were performed to support the analysis in the previous sections. The first set of simulations were conducted with a random choice of the K non-zero coefficients in X and the average probability of error,P_e, curve as a function of the number of non-zero coefficients (sparsity value) K was computed. Here, we define the probability of error of the FFAST algorithm as the fraction of non-recovered non-zero coefficients out of the K non-zero coefficients that were chosen initially. The second set of simulations were conducted for busty signal with two equal bursts of length $K/2$, separated by a separation, s, and the plot for average probability of error, P_e, averaged over all possible separations, s, $0 \leq s \leq N - 2K$, was generated as a function of K. The plots for $N = 250 \times 251$ are shown in the Fig. 7.

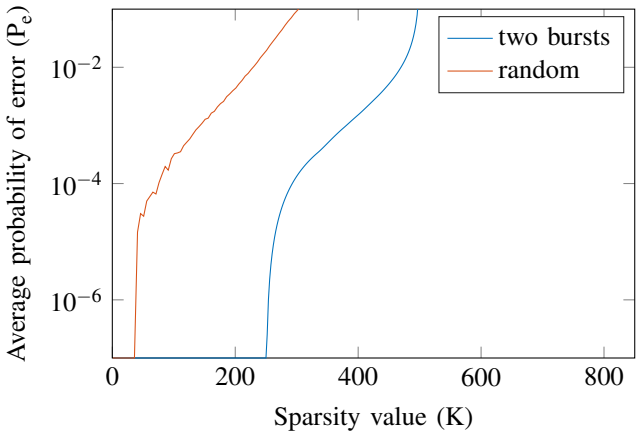

Fig. 7. Plot of the average probability of error , P_e, as a function of the sparsity value, K, for bursty and random selection of non-zero coefficients (N = 250*251 and 1002 samples)

It can be seen from the plot that for the same sparsity level, the performance of the FFAST algorithm is better for the two burst case. It can also be seen that even for the $d = 2$ case, there is a sharp threshold when there are two bursts and the threshold as seen from the simulations matches exactly with that in the conjecture in the previous section. This should be contrasted with the case of randomly chosen non-zero coefficients for which there is no non-trivial threshold for $d = 2$, i.e., there always exists an error floor even when $N \to \infty$.

References

[1] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete fourier transform using at most $4k$ samples and $o(klogk)$ complexity," in *Proc. IEEE Symposium on Information Theory*, 2013, pp. 464–468.

[2] J. Justesen and T. Høholdt, "Analysis of iterated hard decision decoding of product codes with reed-solomon component codes," in *Information Theory Workshop, 2007. ITW'07. IEEE.* IEEE, 2007, pp. 174–177.

[3] A. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse fourier transform: A compressed fourier transform for big data," *IEEE Signal Processing Mag.*, vol. 31, no. 5, pp. 91–100, 2014.

[4] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Near-optimal algorithm for sparse fourier transform," in *Proc. 44th Annu. ACM Symp. Theory of Computing (STOC)*, 2012.

[5] S. Pawar and K. Ramchandran, "A FFAST framework for computing a k-sparse dft in $o(klogk)$ time using sparse-graph alias codes," http://arxiv.org/pdf/1305.0870.pdf, 2013.

[6] J. Justesen, "Performance of product codes and related structures with iterated decoding," *IEEE Trans. on Commun.*, vol. 59, no. 2, pp. 407–415, 2011.

[7] H. Burton and J. Weldon, E., "Cyclic product codes," *Information Theory, IEEE Transactions on*, vol. 11, no. 3, pp. 433–439, Jul 1965.