EE5160 - Project 2
Fast Sparse 2D-DFT using Sparse-Graph Codes

Nived Rajaraman , EE14B040 Aniruddh V , EE14B008

***Abstract*—This document deals with computing the sparse 2D-FFT of a spatial domain measurement with lower sample and computational complexity as seen compared to a normal 2D-FFT. The algorithm uses the concepts of aliasing, down-sampling and the use of the peeling Decoder to extract the DFT in *sub-linear* time. Given a $N_x \times N_y$ matrix, having k uniformly distributed non-zero DFT coefficients, the 2d-FFAST (Fast Fourier Aliasing based Sparse Transform) algorithm computes the DFT with a sample complexity of $O(logk)$ measurements and $O(klogk)$ computations. We also present a few examples to illustrate the results for different densities (i.e) $\delta < 1/3$ and for $\delta > 1/3$.**

I. Introduction

Spectral analysis has been of great importance in many fields. There is a need for algorithms that can perform real-time, low power signal analysis in devices. The normal FFT algorithm uses $\mathcal{O}(NlogN)$ computations to obtain the spectral waveform. But many imaging applications such as MRIs and astronomical imaging, have a sparse representation in the Fourier domain. Denoting the number of non-zeros coefficients of the sparse DFT by k, with $k \propto N^{\delta}$ (with $0 < \delta < 1$), we can take advantage of this to reduce the computational and sample complexity of the problem. We illustrate an algorithm that extends the 1D-FFAST architecture to 2 dimensions. The FFAST algorithm uses sub-sampling and a Chinese remainder Theorem based mapping to get a sparse graph which is then solved iteratively using the Peeling Decoder. Though we assume no noise and a uniform distribution of spectral coefficients (exactly k-sparse transforms), the algorithm also works with sub-linear complexity for noisy and non-uniform distributions (approximately k-sparse transforms). The algorithm is probabilistic with the probability of error asymptotically reaching 1.

II. Problem Formulation

Consider an exactly sparse 2D signal $\mathbb{S}$ with column and row dimensions N_1 and N_2. The overall signal dimension $N = N_1 N_2$. In addition, the algorithm requires that N_1 and N_2 can be factored into $\{Q_i^{1,2}\}_{i=0}^{d-1}$ with $\{Q_i^1 Q_i^2\}_{i=0}^{d-1} = \{Q_i\}_{i=0}^{d-1}$, where d is an appropriately chosen constant (usually chosen as 3). Each of the factors of the overall dimension $\{Q_i\}$ are of the order of $N^{\frac{1}{d}}$ and are all pairwise coprime.

$$N = Q_0 Q_1 ... Q_{d-1}$$

The DFT of this signal is assumed to be sparse, with k of the DFT coefficients being non-zero. The non-zero coefficients are assumed to be uniformly distributed with the degree of sparsity quantified by the parameter δ as $\delta = log_N k$. The problem has been segmented into two cases:

A. N_1 and N_2 are coprime to each other

The condition listed above is more relaxed for the case where the ambient signal dimensions are coprime (i.e. $gcd(N1, N2) = 1$) in that the condition on the individual dimensions being factorizable each into d factors is relaxed and only the constraint that $N = Q_0 Q_1 ... Q_{d-1}$ is applicable.

B. N_1 and N_2 are not coprime to each other

The constraints mentioned initially must be satisfied for the more general case of non-coprime dimensions. This limits the practical set of signal dimensions allowed, but often the input size can be trimmed to make it compatible with the 2D-FFAST architecture.

III. Limitations

- The 2D-FFAST architecture does not apply to arbitrary 2D signal dimensions but only to a constrained set of dimensions that can be expressed as illustrated in section II. However, as mentioned previously, the input signal dimensions can be trimmed to satisfy the 2D-FFAST requirement.
- The algorithm being probabilistic generates the correct result with a probability that goes to 1 asymptotically in k (k being sub-linear in the ambient signal dimension N).
- The obtained analytical results also assume a uniformly random model for the support of the non-zero DFT coefficients, which does not describe real-world signal accurately.

IV. 2D-FFAST - coprime signal dimensions

This case is an extension of the 1D-FFAST architecture. The 2D-FFAST with coprime dimensions can be reduced to a 1D-FFAST using a unique forward mapping based on the Chinese Remainder Theorem and can be regenerated from the 1D-DFT using a unique reverse mapping - the Good Thomas map.

A. Forward Mapping (Chinese Remainder Theorem)

The mapping from the 2D signal to the 1D signal is accomplished using a mapping based on the Chinese Remainder Theorem. The constraint that the signal dimensions need be coprime is enforced while performing this mapping. The Chinese Remainder Theorem informally states that for any integer N with coprime factors $n_0, n_1, n_2 ... n_{d-1}$, any integer from $[0, N-1]$ can be uniquely determined by its remainders with each of the factors. The implication of this is that since

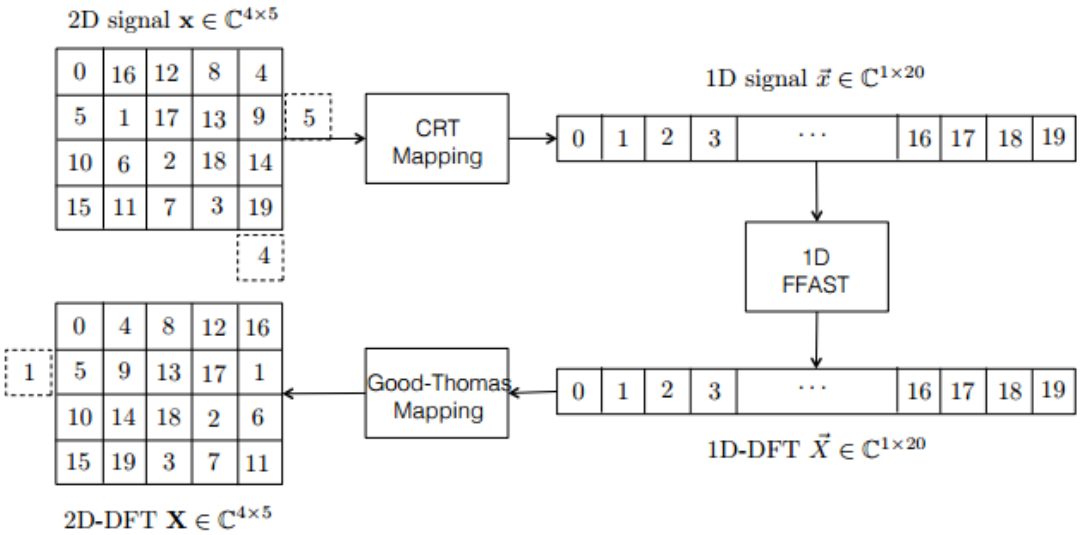

Fig. 1: Forward and Reverse Mapping of the 2D-signal to and from a 1D-signal (coprime dimensions)

we can deconstruct N into two coprime factors N_1 and N_2, the Chinese remainder theorem assures that any point in the 1D-DFT $x_i \in [0, N-1]$ can be uniquely represented in terms of:

$$a_1 = x_i \pmod{N_1}, \qquad a_1 \in [0, N_1 - 1]$$
$$a_2 = x_i \pmod{N_2}, \qquad a_2 \in [0, N_2 - 1]$$

. Given a_1 and a_2, x_i can be reconstructed as:

$$x_i = a_1 N_2 N_2^{-1} + a_2 N_1 N_1^{-1}$$
$$Where\ N_{1,2}^{-1} \ :\ N_{1,2}^{-1} N_2 \equiv 1 \pmod{N_{2,1}}$$

Thus for the forward mapping any point of the input signal $(n_{1,i}, n_{2,i})$ can be done taking it as (a_1, a_2) and mapping it to a point on the 1D-signal x_i using the Chinese remainder theorem as shown above. The Theorem guarantees such a mapping is bijective (and hence unique and invertible).

B. Reverse Mapping (Good Thomas Algorithm)

Once the 1D-DFT is computed, the coefficients need to be mapped back to the order corresponding to that of the original 2D-DFT (since the forward mapping rearranges the order). This is achieved using the reverse mapping based on the Good Thomas Algorithm. Consider a point in the 1D-signal (x). This can be mapped back to a unique point in the 2D-signal (n_1, n_2) that satisfies:

$$x_i = (n_{2,i} N_1 + n_{1,i} N_2) \pmod{N}$$

It can be shown that $n_{1,i}$ and $n_{2,i}$ are unique modulo N_1 and N_2 respectively and hence all the $N = N_1 N_2$ points in the 1D-transform can be uniquely mapped back to the 2D-transform. $n_{1,i}$ and $n_{2,i}$ can be trivially solved for as:

$$n_{1,i} = x_i \pmod{N_2}$$

$$n_{2,i} = x_i \pmod{N_1}$$

Note that the requirement that N_1 and N_2 is implicit in order to be able to uniquely map from $(n_{1,i}, n_{2,i})$ to x_i and vice versa.

There are several bijective maps from a 2D to a 1D-signal (and vice versa). The reason for choosing the two aforementioned maps [] is solely for the reason that together they generate a *permutation* of the 2D-DFT and hence there is no additional computational overhead (besides the mapping itself) in the form of having to perform any linear separation operations on the 1D-DFT.

C. Subsampling

Once the 2D-DFT has been mapped to the 1D-DFT, a subsampling operation is performed several times on the signal to generate a smaller dimension signal. The rates of subsampling chosen are determined as portayed later on. The Peeling Decoder also requires that the subsampling operation be performed on D delayed (circular shifted) versions of the input signal, with the delay chosen as 1. In our implementation, a single 1-delayed signal is subsampled along with the undelayed signal for the Peeling Decoder to function (for each subsampling rate).

It is here that the notation of **streams** and **sub-streams** is introduced. Each version of the DFT of the subsampled signal and its delayed copies at a particular rate s corresponds to a stream, denoted by S_s; while each the DFT of each subsampled delayed copy corresponds to a sub-stream notated $\widehat{X}_{s,d}$, with d being the delay ($\in [0, D-1]$. The subsampled coefficients can be expressed in terms of the coefficients of the original DFT, $\widehat{X}$ as a simple linear combination. For the undelayed coefficient:

$$\widehat{X}_{s,0}[j] = \widehat{X}[j] + \widehat{X}[N/R + j] + ... + \widehat{X}[N - N/R + j]$$

$$\widehat{X}_{s,0}[j] = \Sigma_{t=0}^{t=R-1} \widehat{X}[j + t.N/R] \quad (1)$$

For the delayed coefficients:

$$\widehat{X}_{s,d}[j] = \Sigma_{t=0}^{t=R-1} \widehat{X}[j + t.N/R] e^{\frac{2\pi d}{N}(j+t.N/R)} \quad (2)$$

The subsampling operation is motivated strongly by 2 aspects:

- Reduced complexity of computing the Fourier Transform
- The ability to locate the position of the non-zero DFT coefficients efficiently (appears later on in the Peeling Decoder).

The first aspect is seen from the fact that the Fourier transform itself has complexity $\mathcal{O}(NlogN)$ but subsampling and computing the (partial) coefficients would take $\mathcal{O}(\frac{N}{d} \log \frac{N}{d} \times d) = \mathcal{O}(N \log \frac{N}{d})$. Of course these do not represent the actual coefficients (and hence have been labelled textitpartial), but the Peeling Decoder elegantly brings it all together, enabling both the **locations** as well as the **values** of the sparse coefficients to be computed in sublinear time, which is the essence of the second aspect.

Coming to the actual implementation, the subsampling is performed at different rates chosen based on the ambient signal dimension N. A condition mentioned earlier was the factorizability of the total dimension N into d factors $Q_1, Q_2...Q_{d-1}$ of the same order. Optimally the subsampling rates are chosen as: $\frac{N}{Q_1}, \frac{N}{Q_2}...\frac{N}{Q_{d-1}}$. Note that there are no hard constraints on the rates of subsampling and the performance varies as the chosen values deviate.

D. Peeling Decoder

The Peeling Decoder plays a crucial role in reducing the computational complexity in this approach to the sparse DFT problem, by allowing the subsampled DFT coefficients to be efficiently reconstructed to give the original DFT values (henceforth referred to as the reconstruction problem). The Peeling Decoder runs with a complexity of $\mathcal{O}(k)$, linear in the number of sparse DFT coefficients. This is detailed further ahead.

The Peeling Decoder as the name suggests follows an onion peeling like approach to the reconstruction problem. The Decoder iteratively computes what are referred to as *singleton nodes* and uses these to solve for the DFT coefficients. The Decoder's success relies on the subsampling rates being chosen appropriately (taking the sparsity index δ into consideration). The working of the Decoder is as portrayed below. Firstly, a Tanner graph is constructed that links together the subsampled DFT coefficients with those of the original Decoder as per equation 1. The DFT being sparse means that there is likelihood that some of the subsampled DFT coefficients (called check nodes) are expressed in terms of a single non-zero DFT coefficient. Such check nodes are termed as *singleton nodes*. Considering equations 1 and 2, we see that given that a check node is singleton, the phase with respect to its 1-delayed version ($d = 1$) is an integral multiple of $\theta = -\frac{2\pi}{N}k$, k being the support of the non-zero coefficients. Though the statement may not be true the other way around, it can be shown that depending on the sparseness this test for a check node being singleton gives a low probability of false positive that asymptotically (in k) goes to 0. This test is known as the **singleton test** and can be used to ascertain whether a check node is singleton or not - first find $p = \frac{2\pi}{N} \angle \widehat{X}_{s,1}[j] / \widehat{X}_{s,0}[j]$; if it is an integer, there is a high probability that the node is singleton corresponding to the DFT coefficient at p of value $\widehat{X}_{s,0}[j]$.

With this in place, we define check nodes as being zerotons and multitons with 0 and more than 1 non-zero DFT coefficients contributing to the check node respectively. The Peeling Decoder can be understood through a series of steps as listed below:

- Iterate through the list of check nodes and using the singleton test judge if any nodes are singleton.
- If no singleton check nodes are found and all check nodes have been made 0, terminate execution. If non-zero check nodes exist, the Peeling Decoder has failed.
- If a singleton node is found, update its value into the final DFT estimate in the position (node) given by the singleton test.
- Delete the all the edges of the Tanner graph connected to that node as well as its singleton check node(s). Subtract the contribution of the deleted node from all the check nodes it occurs in.
- Repeat from step 1

E. Performance

The 2D-FFAST architecture for coprime dimensions is an extension of the 1D-FFAST. The performance guarantees thus hold for the 2D-FFAST as there is no overhead of having to perform the CRT and Good Thomas mapping. By re-indexing the DFT expressions in terms of the mapped coefficients, there would be no requirement to rearrange the signal (and coefficients) in memory (which has a complexity of $\mathcal{O}(N)$). The Peeling Decoder being sublinear does not make its way into the complexity expression and thus we are left with the *reduced complexity* DFT (FFT) which as shown before is of the order of $\mathcal{O}(k \log k)$).

As far as the sample complexity is concerned, the number of measurements that would have to be made is of the order of $\mathcal{O}(k)$. This can be seen from the fact that the 2D-DFT can be trivially mapped to a 1D-FFT which guarantees an $\mathcal{O}(k)$ measurement complexity.

V. 2D-FFAST - NON-COPRIME SIGNAL DIMENSIONS

The case of non-coprime signal dimensions requires the 1D-FFAST to be adapted to the 2D case. It can be shown that performing a 2D-FFAST on a signal with non-coprime dimensions can be reduced to a 1D-FFAST but the mapping is not as trivial as in the coprime case. In this case, we directly subsample the signal along both dimensions and use a modified

Peeling Decoder to regenerate the original DFT coefficients. Note that as we are always working in the 2D domain, there is no forward or reverse mapping as in the coprime dimension case.

A. Subsampling

The subsampling operation performed in the case of the 2D-FFAST is virtually the same as in the 1D case, simply extended to 2 dimensions. As mentioned in section II, (N_1, N_2) should be factorizable into factors $\{Q_i^{1,2}\}_{i=0}^{d-1}$ and these are the chosen rates for subsampling. The constraint that $\{Q_i\}_{i=0}^{d-1} = \{Q_i^{1,2}\}_{i=0}^{d-1}$ are of the same order implies that the sizes of the subsampled signal would be of the same order. As in the 1D-FFAST, the signal is also subsampled after delaying it D_1 and D_2 times along either dimension. In the exactly k-sparse case, it is sufficient to take $D_1 = D_2 = 1$. The notation of streams and sub-streams and the signal notation is continued as used before (only difference being the double indexing).

$$\widehat{X}_{s,0}[i,j] = \Sigma_{t_1=0}^{R_1-1}\Sigma_{t_2=0}^{R_2-1}\widehat{X}[i+t_1.N_1/R_1, j+t_2.N_2/R_2] \quad (3)$$

For the delayed coefficients:

$$\widehat{X}_{s,(d_1,d_2)}[i,j] = \Sigma_{t_1=0}^{R_1-1}\Sigma_{t_2=0}^{R_2-1}\widehat{X}[c_1,c_2]e^{\frac{2\pi d_1}{N_1}c_1}e^{\frac{2\pi d_2}{N_2}c_2} \quad (4)$$

$$Where \;\; c_1 = (j + t_2.N/R_2) \;,\; c_2 = (i + t_1.N/R_1)$$

For the exactly k-sparse case, we only require 2 delayed copies of the signal to be subsampled, once vertically and once horizontally. These correspond to $\widehat{X}_{s,(1,0)}$ and $\widehat{X}_{s,(0,1)}$

Similar to the previous case, we can compute the row and column support of a node corresponding to a singleton node by performing the singleton test on $\widehat{X}_{s,(0,0)}[i,j]$, $\widehat{X}_{s,(1,0)}[i,j]$ and $\widehat{X}_{s,(0,1)}[i,j]$, which if successful would give:

$$\begin{aligned}
\text{value} &= \widehat{X}_{s,(0,0)}[i,j] \\
\text{column support} &= \frac{N_1}{2\pi}\angle\widehat{X}_{s,(1,0)}[i,j]/\widehat{X}_{s,(0,0)}[i,j] \\
\text{row support} &= \frac{N_2}{2\pi}\angle\widehat{X}_{s,(0,1)}[i,j]/\widehat{X}_{s,(0,0)}[i,j]
\end{aligned}$$

Thus for ease of clarity and implementation we construct an object (bin) for every element in a stream denoted $Y_{s,i,j}$ with s being the subsampling rate.

$$Y_{s,i,j} = \begin{pmatrix} \widehat{X}_{s,(0,0)}[i,j] \\ \widehat{X}_{s,(1,0)}[i,j] \\ \widehat{X}_{s,(0,1)}[i,j] \end{pmatrix}$$

B. Peeling Decoder

The Peeling Decoder used in the case of the 2D-FFAST for non-coprime dimensions is a slightly modified version of the previously used Peeling Decoder. The difference here is that the modified singleton test is performed on the bins to that outputs the row, column and value estimate of the node (corresponding to that check node) and the computed value must be subtracted from the other (doubly indexed) check nodes to which that node contributes. It can be shown that every non-zero DFT coefficient contributes to exactly 1 bin per stream the indices of which can be computed based on the row (row) and column (col) support of the DFT coefficient:

$$n_{1,i} = col \pmod{N_1/Q_{1,i}}$$

$$n_{2,i} = row \pmod{N_2/Q_{2,i}}$$

$Q_{1,i}$ and $Q_{2,i}$ are the column and row subsampling rates

C. Performance

As mentioned before, it can be shown that the 2D-FFAST architecture can always be non-trivially mapped to a 1D-FFAST. This allows the performance guarantees of the 1D-FFAST to apply to its 2D counterpart without alteration. Thus the sample complexity of the 2D-FFAST is $\mathcal{O}(k)$ while the computational complexity is $\mathcal{O}(k \log k)$.

VI. Test Case

Here we consider a low dimensional test case for the 2D-FFAST with unequal, non-coprime dimensions $(N_1, N_2) = (304, 306)$. The number of non-zero DFT coefficients is k is 4 (small value chosen for illustration purposes) at positions (values in subscripts): $(2,2)_{10}, (226,97)_{-23}, (150,300)_1, (36,154)_{1.5}$. The 2D-FFAST implementation for this case is explained below:

Subsampling

Two streams have been chosen for subsampling and the rates for the column and row dimensions are chosen as $(16, 18) = 288$ and $(17, 19) = 323$ which are of the same order. The dimensions of the subsampled signal are then - $(19, 17)$ and $(18, 16)$ respectively. As mentioned before we consider 3 sub-streams corresponding to the undelayed, 1-delayed (column) and 1-delayed (row) signals.

Peeling Decoder

Below we demonstrate the Tanner graph on which the Peeling Decoder operates. Since there are only 4 non-zero coefficients, the graph has only 4 left-nodes while the number of check nodes would be $19 \times 17 + 18 \times 16 = 611$. The Tanner graph corresponding to the Peeling Decoder has been illustrated further below. The check nodes in dark gray are the multiton nodes, those in light gray are the singleton nodes and those in white correspond to the zeroton check nodes.

From the diagram, we can see that the Peeling Decoder iteratively solves for the DFT coefficients in the order

$$(150, 300) \rightarrow (36, 154) \rightarrow (2, 2) \rightarrow (226, 92)$$

Note that most of the check nodes (being 0) would be evaluated to zerotons (or possibly multitons) and the 6 nodes mentioned would be the only ones that trigger giving non-zero values.

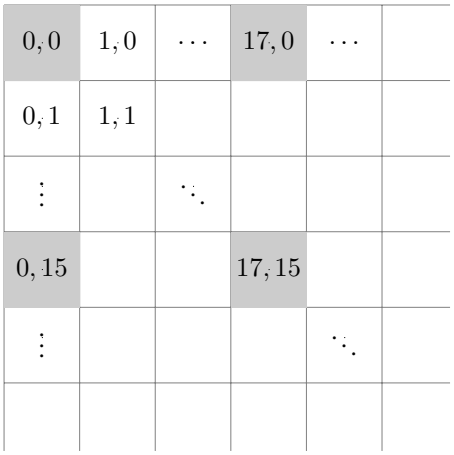

Fig. 2: 2D-signal coefficients with subsample indices in gray

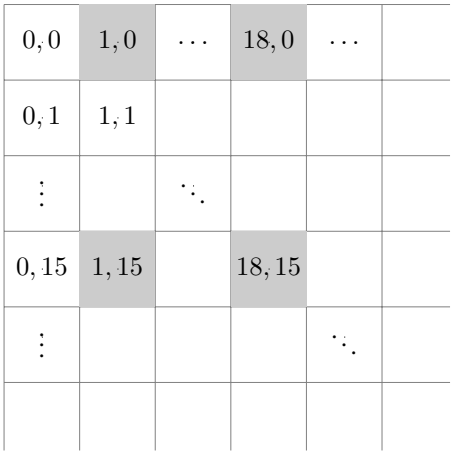

Fig. 3: x-delayed 2D-signal with subsample indices in gray

VII. Analytical and Empirical Results

In the following, we discuss the complexity of the 2D-FFAST algorithm considering each of the front-end and back-end processing steps. The complexities which affect the performance of the algorithms are the sample complexity which is the number of measurements that need to be made to compute the DFT. This gives a measure of the number of reads made to time domain matrix. The other benchmark is the computational complexity, which is a measure of the time to compute through all the steps in the algorithm.

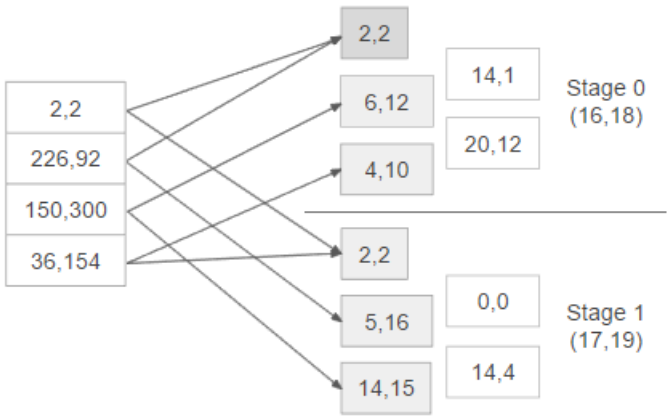

Fig. 4: Tanner Graph corresponding to the Peeling Decoder

Subsampling

Consider a signal of total ambient dimension N, and sampling rates f_0, f_1, $\ldots$, f_{d-1}. For the very-sparse regime of $0 < \delta \leq 1/3$, the 2D-FFAST architecture with $d = 3$ stages is sufficient to compute a k-sparse length DFT using m = 2.44k samples. For the less-sparse regime of $1/3 < \delta \leq 0.99$, the number of stages can vary based on the value of δ (typically from 6 to 8). The total samples used for these regions are at max m = 3.74k. In general for any fixed $0 < \delta < 1$, the sample complexity m can be as small as $r.k$, where r is the oversampling constant, explained below. Since k cannot be exactly determined beforehand, the number of stages cannot be determined dynamically and for safe functioning, the number of stages would have to be chosen such that they should always be capable of solving for more than k coefficients. This is quantified in terms of the oversampling constant r, which is the ratio of the samples taken to k. Hence, the sampling complexity is $m = O(k)$.

DFT computation

The 2D-FFAST algorithm then computes $3.d$ DFTs (d stages and 3 delays in each stage), each of length approximately equal to $l = \eta k$ (where the threshold η is the average number of bins per stage, normalized by the sparsity). For the very-sparse regime of $0 < \delta \leq 1/3$, the 2D-FFAST algorithm can obtain the DFT for all $\eta > 0.38$. For the less-sparse regime of $1/3 < \delta \leq 0.99$, the algorithm can obtain the DFT for all $\eta > 0.47$. Since an N-point FFT has $O(N \log N)$ time complexity, the computation of 3d DFTs will take $O(k \log k)$ time as the value of η and d does not vary significantly with N and sparsity δ. Hence the total computational complexity is $O(k \log k)$

Peeling Decoder

Solving a sparse graph using a peeling Decoder has a time complexity that is of the order of the number of edges in the associated Tanner graph. As the number of bit nodes are k,

and each bit node has one edge to a check node in each of the stages (as shown in section V), the total number of edges in the Tanner graph is $d.k$. Hence, the complexity of the Peeling Decoder is $O(k)$.

Comparison

We see from the previous sections that the overall complexity of the 2D-FFAST algorithm is $O(k \log k)$ for the computation of the DFT. This complexity is sublinear as it is less than $O(N)$. Let us compare the complexity of with an example. Considering an image of resolution 1366 $\times$ 768, we get $N = 10^6$. Taking a sparsity of $k = 1000$ (which falls in the less sparse regime), extending the asymptotic complexity to this finite input size case, we see a gain of 500 times. As the size of the time domain matrix increases, the behaviour converges to the asymptotic complexity and the gain becomes more prominent as well.

VIII. Conclusion

In this document, the FFAST algorithm was introduced to compute the 2D-FFT of a signal having a k-sparse Fourier Transform. We also showed its implementation using aliasing due to subsampling and the usage of the Peeling Decoder. The performance of the algorithm was discussed and shown to be sublinear as $O(klogk)$. Though the document was for exactly k-sparse, the algorithm can also be extended to noisy signals with high SNR. In such cases, the complexity for reliable DFT computation would increase, but would still remain sublinear.

Algorithm	2D-FFAST	2D-R-FFAST
Noise model	Noiseless	Noisy with sufficiently high SNR
Sample complexity	$O(k)$	$O(k \log^3 N)$
Computational complexity	$O(k \log k)$	$O(k \log^4 N)$

Fig. 5: Summary of Complexity

References

[1] Frank Ong, Sameer Pawar, and Kannan Ramchandran, *Fast and Efficient Sparse 2D Discrete Fourier Transform using Sparse-Graph Codes*, IEEE, 2015

[2] Sameer Pawar, and Kannan Ramchandran, *Computing a k-sparse n-length Discrete Fourier Transform using at most 4k samples and O(k log k) complexity*, IEEE, 2013

[3] I. J. Good, *The interaction algorithm and practical Fourier analysis*, Journal of the Royal Statistical Society. Series B (Methodological), 1958

[4] R. Tolimieri, Myoung An, and Chao Lu, *Good-Thomas PFA* Springer, 1989, pp.119-120

[5] RS Stankovic, and MS Stankovic, *Remarks on History of FFT and Related Algorithms*

[6] Krishna R. Narayanan, *The Peeling Decoder : Theory and some Applications* [online] pfister.ee.duke.edu/nasit16/Narayanan.pdf