Alias Codes for Sparse Fourier Transforms

Kannan Ramchandran

Joint Work with Sameer Pawar and Xiao (Simon) Li
UC Berkeley
FOCS 2014 Workshop on The Sparse Fourier Transform Theory and Applications, Pennsylvania
Acknowledgements

- Frank Ong
- Quentin Byron
- Thibault Derousseaux
- Orhan Ocal
Acknowledgements

Piotr Indyk and Dina Katabi
Outline

• Part I: Noiseless Recovery
  – Sparse-Graph Alias Codes

• Part II: Noisy Recovery
  – Sample-Optimal Recovery with Near Linear Run-time
  – Near Sample-Optimal Recovery with Sub-linear Run-time
PART I: Noiseless Recovery
Computing Sparse DFT | Problem Formulation

• Compute the $K$-sparse DFT of $\mathbf{x} \in \mathbb{C}^N$ with $K \ll N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k \in \mathcal{K}} X[k] e^{i \frac{2 \pi k}{N} n}, \quad n = 0, \ldots, N - 1$$

$\mathcal{K} = \text{chosen from } [N] \text{ uniformly at random}$

• Classical solution: FFT algorithm
  - Sample cost = $N$
  - Computational cost = $\mathcal{O}(N \log N)$

• With sparsity, what are the fundamental bounds for support recovery?
  - Sample cost?
  - Computational cost?
Computing Sparse DFT | Numerical Phantoms for Cardiovascular MR

http://www.biomed.ee.ethz.ch/research/bioimaging/cardiac/mrxcat

336 = 16 \times 21
323 = 17 \times 19
temporal difference across different frames of the phantom
Computing Sparse DFT | Numerical Phantoms for Cardiovascular MR
Computing Sparse DFT | Problem Formulation

• Compute the $K$-sparse DFT of $x \in \mathbb{C}^N$ with $K \ll N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k \in \mathcal{K}} X[k] e^{i \frac{2\pi k n}{N}} + w[n], \quad n = 0, \ldots, N - 1$$

$\mathcal{K}$ = chosen from $[N]$ uniformly at random

Assumptions and caveats:

• $X[k]$ is from a finite constellation.

• Sparsity $K = |\mathcal{K}| = \mathcal{O}(N^\delta)$ is sub-linear for some $\delta \in (0, 1)$.

• Noise $w[n]$ is independent complex Gaussian $\mathcal{CN}(0, \sigma^2)$

• Recovery guarantees are probabilistic.
Computing Sparse DFT | Problem Formulation

- Compute the $K$-sparse DFT of $x \in \mathbb{C}^N$ with $K \ll N$:

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k \in \mathcal{K}} X[k] e^{i \frac{2 \pi k}{N} n} + w[n], \quad n = 0, \ldots, N - 1$$

$$\mathcal{K} = \text{chosen from } [N] \text{ uniformly at random}$$

**Theorem:** FFAST algorithm computes the $K$-sparse DFT of $x \in \mathbb{C}^N$,

- using $M = O(K)$ samples,
- in $O(K \log K)$ computations,
- with probability at least $1 - O(1/M)$. 
Computing Sparse DFT | Related Work

Statistical Signal Processing
- Frequency estimation
  - Prony [1795]
  - Pisarenko [’73]
  - Vetterli et al. [’02]
  - more ⋯
- Subspace methods [’86]
  - MUSIC, ESPRIT etc.

Compressed Sensing
- Feng and Bresler [1996]
- Candes et. al [2006]
- Rauhut [2008]
- Wainwright [2009]
- Tang et. al [2012]
- more ⋯

Sparse DFT
- Gilbert et. al [2002-2008]
- Mishali and Eldar [2010]
- Iwen [2010]
- Indyk et al. [2012]
- more ⋯

SFFT: Sparse Fast Fourier Transform

\[ Y = \Phi x \]

\( M \times 1 \)
\( M \times N (M < N) \)
\( N \times 1 \)

The discrete Fourier transform (DFT) is one of the most important and widely used
More recent work on computing sparse DFT
- [http://groups.csail.mit.edu/netmit/sFFT/paper.html](http://groups.csail.mit.edu/netmit/sFFT/paper.html)

Recent advances in compressed sensing and sketching methods

Our method:
- targets support recovery instead of $\ell_2/\ell_1$ approximation
- uses the design and analysis of sparse-graph codes
- leverages harmonic retrieval methods in statistical signal processing
Computing Sparse DFT | Insights

- Computes an exactly $K$-sparse $N$-length DFT using $O(K)$ samples with $O(K \log K)$ computations.
- A common framework for noiseless and noisy observations.
Computing Sparse DFT | Insights

- sub-sampling below Nyquist rate
- clever sub-sampling (for sparse case)
- Chinese-Remainder-Theorem guided subsampling

Aliasing in the frequency domain

good “alias code”?

Sparse graph codes
Computing Sparse DFT | Insights

- sub-sampling below Nyquist rate
- Aliasing in the frequency domain

- clever sub-sampling (for sparse case)
- good “alias code”? 

Chinese-Remainder-Theorem guided subsampling
- Sparse graph codes

Coding-theoretic tools

- Design:
  - Randomized constructions of good sparse-graph codes.

- Analysis:
  - Density evolution
  - Martingale
  - Expander graph theory
Computing Sparse DFT | Main Idea

**time-domain** $x[n]$ length $N = 20$

**frequency-domain** $X[k]$ sparsity $K = 5$

↓ 5

subsample by 5


Computing Sparse DFT | Main Idea

**time-domain $x[n]$ length $N = 20$**

**frequency-domain $X[k]$ sparsity $K = 5$**

Subsample by 5

\[ X[3] = 4 \quad X[10] = 3 \]
\[ X[1] = 1 \quad X[5] = 1 \]

\[ X[13] = 7 \]

\[ U[0] \quad U[1] \quad U[2] \quad U[3] \]
Computing Sparse DFT | Main Idea

**time-domain** $x[n]$ length $N = 20$

**frequency-domain** $X[k]$ sparsity $K = 5$

↓ 5

subsample by 5


Computing Sparse DFT | Main Idea

**time-domain** $x[n]$  length $N = 20$

**frequency-domain** $X[k]$  sparsity $K = 5$

subsampling by 5

$$
\begin{align*}
X[3] &= 4 \\
X[1] &= 1 \\
X[5] &= 1 \\
X[10] &= 3 \\
X[13] &= 7 \\
\end{align*}
$$

$$
\begin{align*}
\end{align*}
$$
Computing Sparse DFT | Main Idea

Time-domain $x[n]$ length $N = 20$

Frequency-domain $X[k]$ sparsity $K = 5$

Subsample by 5
Computing Sparse DFT | **Main Idea**

time-domain $x[n]$ length $N = 20$

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

frequency-domain $X[k]$ sparsity $K = 5$

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

↓ 5

subsample by 5

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

$X[3] = 4$

$X[10] = 3$

$X[13] = 7$

$X[1] = 1$

$X[5] = 1$

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


zero-ton
Computing Sparse DFT | Main Idea

**time-domain** \( x[n] \) **length** \( N = 20 \)


**frequency-domain** \( X[k] \) **sparsity** \( K = 5 \)

$$X[0] = 0 \quad X[10] = 3 \quad X[13] = 7$$

Subsample by 5

\( \downarrow 5 \)

$$U[0] \quad U[1] \quad U[2] \quad U[3]$$

Zero-ton

\( \Longleftrightarrow \text{DFT} \Rightarrow \)
Computing Sparse DFT | Main Idea

time-domain $x[n]$ length $N = 20$

$X[0] = 1$
$X[3] = 4$
$X[5] = 1$
$X[10] = 3$
$X[13] = 7$

frequency-domain $X[k]$ sparsity $K = 5$

Initialize:

$x[0] = 1$
$x[3] = 4$
$x[5] = 1$
$x[10] = 3$
$x[13] = 7$


downsample by 5

$X[16] = 0$

In frequency-domain:

zero-ton
multi-ton

$U[1]$
$U[2]$
$U[3]$

$\downarrow DFT \iff\$

$\downarrow 5$

subsample by 5

$\downarrow DFT \iff\$

$\downarrow 5$

sequence

$\downarrow DFT \iff\$

$\downarrow 5$

$\downarrow DFT \iff\$

$\downarrow 5$
Computing Sparse DFT | Main Idea

time-domain $x[n]$ length $N = 20$

$$\downarrow 5$$

subsample by 5

frequency-domain $X[k]$ sparsity $K = 5$

$$\Leftrightarrow \text{DFT} \Rightarrow$$


zero-ton multi-ton
Computing Sparse DFT | Main Idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$

↓ 5

subsample by 5

$X[3] = 4$
$X[1] = 1$
$X[5] = 1$
$X[10] = 3$
$X[13] = 7$


zero-ton multi-ton single-ton

$X[0] = 1$
$X[1] = 1$
$X[5] = 1$
$X[10] = 3$
$X[13] = 7$

$U[0] = 1$
$U[1] = 1$
$U[2] = 1$
$U[3] = 1$
Computing Sparse DFT | **Main Idea**

**time-domain** \(x[n]\)  \(\text{length } N = 20\)


\(\downarrow 5\)

**subsampling by 5**

**frequency-domain** \(X[k]\)  \(\text{sparsity } K = 5\)

\(\Leftrightarrow \text{DFT} \Rightarrow \)

\[ U[0] \quad U[1] \quad U[2] \quad U[3] \]

zero-ton  multi-ton  single-ton
Computing Sparse DFT | Main Idea

time-domain $x[n]$ length $N = 20$

frequency-domain $X[k]$ sparsity $K = 5$

subsample by 5

$\downarrow 5$

DFT

$U[0]$ zero-ton
$U[1]$ multi-ton
$U[2]$ single-ton
$U[3]$ single-ton
Computing Sparse DFT | Main Idea

**time-domain** $x[n]$ length $N = 20$

**frequency-domain** $X[k]$ sparsity $K = 5$

$$\downarrow 5$$

subsample by 5

$$\iff DFT \iff$$
Computing Sparse DFT | **Main Idea**

**time-domain** $x[n]$  
length $N = 20$

**frequency-domain** $X[k]$  
sparsity $K = 5$

$$X[1] = 1 \quad X[5] = 1$$
$$X[13] = 7$$

$$\downarrow 5$$

shift & subsample by 5
Computing Sparse DFT | Main Idea

time-domain $x[n]$ length $N = 20$

$$\Downarrow 5$$

shift & subsample by 5

$$\iff \text{DFT} \iff$$

frequency-domain $X[k]$ sparsity $K = 5$

$$\begin{align*}
\end{align*}$$

subscript $U_S$ suggests shift

Our Measurements

$$\iff \text{DFT} \iff$$

$$U_S[0] \quad U_S[1] \quad U_S[2] \quad U_S[3]$$
Computing Sparse DFT | Main Idea

\[ x[n] \text{ length } N = 20 \]

\[ X[k] \text{ sparsity } K = 5 \]

\[ \downarrow 5 \]

shift & subsample by 5

\[ \begin{align*}
    X[1] &= 1 \\
    X[3] &= 4 \\
    X[5] &= 1 \\
    X[10] &= 3 \\
    X[13] &= 7
\end{align*} \]

\[ \begin{align*}
    X[0] \times e^{-i \frac{2\pi}{20}} & \quad X[1] \\
    X[3] \times e^{-i \frac{2\pi \times 3}{20}} & \quad X[5] \\
    X[10] \times e^{-i \frac{2\pi \times 10}{20}} & \quad X[13]
\end{align*} \]

\[ \leftrightarrow \text{DFT} \leftrightarrow \]

\[ \begin{align*}
    U_S[0] & \quad \text{zero-ton} \\
    U_S[1] & \quad \text{multi-ton} \\
    U_S[2] & \quad \text{single-ton} \\
    U_S[3] & \quad \text{single-ton}
\end{align*} \]
Computing Sparse DFT | **Main Idea**

subsample by 5

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

**DFT**


zero-ton multi-ton single-ton single-ton


\[ \times e^{-i \frac{2\pi}{20}} \quad \times e^{-i \frac{2\pi \times 3}{20}} \quad \times e^{-i \frac{2\pi \times 5}{20}} \quad \times e^{-i \frac{2\pi \times 10}{20}} \quad \times e^{-i \frac{2\pi \times 13}{20}} \]


shift & subsample by 5

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

**DFT**

\[ U_s[0] \quad U_s[1] \quad U_s[2] \quad U_s[3] \]

zero-ton multi-ton single-ton single-ton
Why do we need shifts?

Computing Sparse DFT | Main Idea

Stage 1

downsampling by 5

\[ X[13] \]
\[ X[10] \]
\[ X[5] \]
\[ X[3] \]
\[ X[1] \]

\[ U[0] \]
\[ U[1] \]
\[ U[2] \]
\[ U[3] \]

DFT ➔ \[ x[n] \]
Computing Sparse DFT | Main Idea

Stage 1
downsampling by 5

$X[13]$
$X[10]$
$X[5]$
$X[3]$
$X[1]$

$U[0], U_S[0]$
$U[1], U_S[1]$
$U[2], U_S[2]$
$U[3], U_S[3]$

Stage 1

DFT $\downarrow$

$\text{shift}$

$x[n]$
Computing Sparse DFT | Main Idea

Stage 1
downsampling by 5

Stage 2
downsampling by 4

X[13]  
X[10]  
X[5]  
X[3]  
X[1]

downsample by 5

downsample by 4

DFT

DFT

DFT

DFT

shift

x[n]

multi-ton

zero-ton

single-ton

multi-ton

zero-ton

single-ton

multi-ton

single-ton

multi-ton
Computing Sparse DFT | Main Idea

Stage 1
downsample by 5

- single-ton
- multi-ton
- zero-ton

Stage 2
downsample by 4

- single-ton
- multi-ton
- zero-ton

Stage 1

Stage 2
Computing Sparse DFT | **Main Idea**

Stage 1  
downsampling by 5

- Single-ton
- Multi-ton
- Zero-ton

Stage 2  
downsampling by 4

- Single-ton
- Multi-ton
- Zero-ton

Stage \( d \)

- Single-ton
- Multi-ton

DFT \( \downarrow \)  

Shift
Computing Sparse DFT | Main Idea

Stage 1
downsampling by 5

Stage 2
downsampling by 4

X[13]
X[10]
X[5]
X[3]
X[1]

single-ton
multi-ton
zero-ton
multi-ton
zero-ton
multi-ton

Zero-ton

Multi-ton

Single-ton

Peeling decoder
Computing Sparse DFT | Main Idea

**Stage 1**
*downsample by 5*

- $X[13]$  
- $X[10]$  
- $X[5]$  
- $X[3]$  
- $X[1]$

**Stage 2**
*downsample by 4*

- single-ton  
- single-ton  
- multi-ton  
- zero-ton  
- zero-ton

**decoder**

peeling
Computing Sparse DFT | Main Idea

Stage 1
downsample by 5

Stage 2
downsample by 4

peeling decoder
Computing Sparse DFT | **Main Idea**

**Stage 1**

downsampling by 5

$X[13]$

**Stage 2**

downsampling by 4

$X[10]$

$X[5]$

$X[3]$

$X[1]$

|------------|-----------|----------|-----------|----------|------------|------------|

peeling decoder
Computing Sparse DFT | Main Idea

Stage 1
downsampling by 5

$X[13]$

Stage 2
downsampling by 4

$X[10]$  
$X[5]$  
$X[3]$  
$X[1]$

peeling decoder
Computing Sparse DFT | **Main Idea**

Stage 1
downsample by 5

Stage 2
downsampling by 4

X[13]  
X[10]  
X[5]   
X[3]   
X[1]   

single-ton  
zero-ton  
zero-ton  
single-ton  
zero-ton  

peeling decoder
Computing Sparse DFT | Main Idea

Stage 1
downsamp by 5

Stage 2
downsamp by 4

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

zero-ton

How to induce good graphs that will work?

peeling decoder
Sparse DFT Computation = Decoding over Sparse Graphs

- **Explicit graph**: design well-understood.
- $(N - K)$ correctly received packets.
- $K$ erased packets.
- Peeling decoder recovers values.

- **Implicit graph** induced by careful sub-sampling
- $(N - K)$ zero DFT coefficients.
- $K$ unknown non-zero DFT coefficients.
- Peeling decoder recovers values & locations.
 CRT-guided Subsampling Induces Good Graphs

- **Balls-and-Bins Model** in Sparse-Graph Codes
- **Chinese-Remainder-Theorem** induced graph

**Chinese-Remainder-Theorem:**
A number between 0-19 is uniquely represented by its remainders modulo (4,5).

- Two graph ensembles are equivalent.
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2:
  - Stage 3:
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3:
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51

49 \times 50 \times 51

49

50

51
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51
- $\delta = 2/3$ such that $K \approx 2500$
  - Stage 1: subsample by $49 \times 50$, keep 51
  - Stage 2: subsample by $50 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 51$, keep 49
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51

- $\delta = 2/3$ such that $K \approx 2500$
  - Stage 1:
  - Stage 2:
  - Stage 3:
More on Subsampling | Choosing Sub-sampling Patterns

Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51

- $\delta = 2/3$ such that $K \approx 2500$
  - Stage 1: subsample by 49, keep $50 \times 51$
  - Stage 2: 
  - Stage 3:
Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51

- $\delta = 2/3$ such that $K \approx 2500$
  - Stage 1: subsample by 49, keep $50 \times 51$
  - Stage 2: subsample by 50, keep $49 \times 51$
  - Stage 3:
Chinese Remainder Theorem

Signal sparsity $K = N^\delta$ with $N = 124950$

- $\delta = 1/3$ such that $K \approx 50$
  - Stage 1: subsample by $50 \times 51$, keep 49
  - Stage 2: subsample by $49 \times 51$, keep 50
  - Stage 3: subsample by $49 \times 50$, keep 51

- $\delta = 2/3$ such that $K \approx 2500$
  - Stage 3: subsample by 51, keep $49 \times 50$

This can be generalized to any sparsity index between 0 and 1
**Goal:** prove that the algorithm finishes $Kd$ steps

$Kd$ edges to be removed
• Pick an arbitrary edge in the graph \((c, v)\).
• Examine its directed neighborhood at depth-2\(\ell\)
• Examine its directed neighborhood at depth-2\(\ell\)
Density Evolution | A Hitchhiker’s Guide

\[ p_\ell = \text{probability of being present at depth-}2\ell \]

\[ \begin{array}{cccc}
  & c & \rightarrow & p_\ell \\
  & v & & \\
  & & p_{\ell-1} & p_{\ell-1} & p_{\ell-1} \\
\end{array} \]
Density Evolution | A Hitchhiker’s Guide

\[ p_\ell = \]

\[ p_\ell = p_{\ell-1} \]

\[ p_\ell = p_{\ell-1} \]

\[ p_\ell = p_{\ell-1} \]

\[ c \]
A Hitchhiker’s Guide

Example with a depth-2 neighborhood

\[ p_\ell = \left[ 1 - (1 - p_{\ell-1})^3 \right] \times \]

In the general setting where each variable has \( d \) edges, the degree distribution of the graph results in the following:

\[ p_\ell \]
\[ p_\ell = \left[ 1 - (1 - p_{\ell-1})^3 \right] \times \left[ 1 - (1 - p_{\ell-1})^2 \right] \times \]

\[ \cdots \]
Density Evolution | A Hitchhiker’s Guide

\[ p_{\ell} = \left[1 - (1 - p_{\ell-1})^3\right] \times \left[1 - (1 - p_{\ell-1})^2\right] \times \left[1 - (1 - p_{\ell-1})^2\right] \]

\( d = 2 \text{ stages} \)

\( \mod 4 \)

\( \mod 5 \)
\( p_\ell = \left[ 1 - (1 - p_{\ell - 1})^3 \right] \times \left[ 1 - (1 - p_{\ell - 1})^2 \right] \times \left[ 1 - (1 - p_{\ell - 1})^2 \right] \)

- It generalizes to \( d \) stages:
Density Evolution | A Hitchhiker’s Guide

\[ p_\ell = \left[ 1 - (1 - p_{\ell - 1})^3 \right] \times \left[ 1 - (1 - p_{\ell - 1})^2 \right] \times \left[ 1 - (1 - p_{\ell - 1})^2 \right] \]

- It generalizes to \( d \) stages:

\[ p_\ell = \left( 1 - e^{-\frac{2Kd}{M}p_{\ell - 1}} \right)^{d-1} \]

- \( K \) = sparsity
- \( M \) = \# of samples
- \( d \) = \# of stages
Density Evolution | A Hitchhiker’s Guide

EXIT Chart

<table>
<thead>
<tr>
<th>$d$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M/2K$</td>
<td>2.0000</td>
<td>1.2219</td>
<td>1.2948</td>
<td>1.4250</td>
<td>1.5696</td>
</tr>
</tbody>
</table>

\[
p_{\ell} = \left(1 - e^{-\frac{2Kd}{M}p_{\ell-1}}\right)^{d-1}
\]

- $K = \text{sparsity}$
- $M = \# \text{ of samples}$
- $d = \# \text{ of stages}$
Sampling Rate | Noiseless Setting: Theory versus Practice

\[ p_\ell = \left(1 - e^{-\frac{2Kd}{M}p_{\ell-1}}\right)^{d-1} \]

- \( N = 7.7 \text{ million} \)
- \( K = 400 \)
- \( d = 3 \text{ stages} \)
- \( M = 1248 \text{ samples} \)

Theoretical threshold = 2 \times 1.23
• Density Evolution
  – assumes that the directed neighborhood is a tree
  – tree-based average analysis

\[ p_\ell = \left(1 - e^{-\frac{2dK}{M} p_{\ell-1}}\right)^{d-1} \]

\( p_\ell \) can be made arbitrarily small
• Density Evolution
  
  – assumes that the directed neighborhood is a tree
  – tree-based average analysis

\[ p_\ell = \left( 1 - e^{-\frac{2dK}{M} p_{\ell-1}} \right)^{d-1} \]

\( p_\ell \) can be made arbitrarily small

\[ Kd(1 - p_\ell) \text{ edges removed} \]
Performance Concentration

- overall average analysis $p^*_\ell$ without tree assumption

$$|p^*_\ell - p_\ell| < \epsilon_1, \forall \epsilon_1 > 0$$

- actual performance concentrates around overall average analysis

$$\mathbb{P}(|\# \text{ of actual remaining edges} - Kd p^*_\ell| > \epsilon_2) \to 0, \quad \forall \epsilon_2 > 0$$

$Kd(1 - p_\ell)$ edges removed
Algorithm Analysis | A Hitchhiker’s Guide

$Kd(1 - p_\ell)$ edges removed

$Kd$ edges to be removed

$Kd p_\ell$ edges remain
• Expander Graph
  - the remaining $Kdp_\ell$ edges form an **expander graph**
  - expander graphs guarantee steady supplies of **single-tons**

  \[ Kd \cdot (1 - p_\ell) \text{ edges removed} \]

  \[ Kdp_\ell \text{ edges remain} \]

  **success** with high probability!
Peeling Performance  | Numerical Examples
Peeling Performance | Numerical Examples

- $N = 600 \times 493 \approx 300$ (thousand)
- $K \approx 25$ (thousand)
- $M \approx 65$ (thousand)
recovered image

remaining image

number of zero/single/multi-tons

count (thousands)
recovered image

remaining image

number of zero/single/multi–tons

count (thousands)
recovered image

remaining image

number of zero/single/multi-tons

count (thousands)
recovered image

remaining image

number of zero/single/multi-tons
recovered image

remaining image

number of zero/single/multi-tons

count (thousands)
recovered image

remaining image

number of zero/single/multi-tons

count (thousands)

0 1 2 3 4 5 6 7
PART II:
Noisy Recovery
Noisy Setting: R-FFAST | Signal Model

- \( y[n] = x[n] + w[n] \), where \( w[n] \in \mathcal{CN}(0, \sigma^2) \).
- There are \( K \) non-zero \( X[k] \), where \( X[k] \) is from a finite constellation.
- SNR is \( \mathbb{E}|x[n]|^2 / \mathbb{E}|w[n]|^2 \) (e.g., SNR=0 dB)
Noisy Setting: **R-FFAST**  |  From Noiseless to Noisy

Noiseless - FFAST

Stage 1

\[ x[n] \]

Stage \( d \)
Noisy Setting: R-FFAST | From Noiseless to Noisy

Noiseless - FFAST

Noisy - R-FFAST

Stage 1

Stage $d$
Noisy Setting: R-FFAST | From Noiseless to Noisy

Two schemes to choose shifts:

- **Scheme 1:**
  - Sample-optimal recovery $M = O(K \log N)$
  - Near-linear run-time $T = O(N \log N)$

- **Scheme 2:**
  - Near sample-optimal recovery $M = O(K \log N)$
  - Sub-linear run-time $M = O(K \log N)$
Noisy Setting: R-FFAST | From Noiseless to Noisy

Two schemes to choose shifts:

- scheme 1:
  - sample-optimal recovery $M = O(K \log N)$
  - near-linear run-time $T = O(N \log N)$
Noisy Setting: R-FFAST | From Noiseless to Noisy

Two schemes to choose shifts:

- scheme 1:
  - sample-optimal recovery
    \[ M = O(K \log N) \]
  - near-linear run-time
    \[ T = O(N \log N) \]

- scheme 2:
  - near sample-optimal recovery
    \[ M = O(K \log^{1.3} N) \]
  - sub-linear run-time
    \[ M = O(K \log^{2.3} N) \]
Architecture | Recap on FFAST Sampling

Each DFT coefficient represents a distinct frequency.
The key is to pinpoint single-tons in terms of

- location
- value

Equivalent to the parameters of a discrete sinusoid

- frequency
- amplitude

\[
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times2} & \cdots & W^{19\times19}
\end{bmatrix}
\]

\[N \times N \text{ DFT matrix}\]

\[W = e^{-i\frac{2\pi}{N}} \text{ with } N = 20\]
Architecture | From Noiseless to Noisy

delay spacing = 1

Noiseless

matching 2 points of a noiseless sinusoid

\[
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times2} & \cdots & W^{19\times19}
\end{bmatrix}
\]

Noisy

delay spacing = random
### Architecture | From Noiseless to Noisy

#### Noiseless

Matching 2 points of a noiseless sinusoid

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19}
\end{pmatrix}
\]

#### Noisy

Matching \(O(\log N)\) random points of a noisy sinusoid

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19}
\end{pmatrix}
\]

delay spacing = 1

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19}
\end{pmatrix}
\]

delay spacing = random

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19}
\end{pmatrix}
\]
Theorem: R-FFAST algorithm computes the $K$-sparse DFT of $x \in \mathbb{C}^N$,
- using $M$ noisy samples, where $M = O(K \log N)$, \textbf{(order optimal)}
- with probability at least $1 - O(1/M)$,
- in $O(N \log N)$ computations.
- for a finite SNR.

Theorem: FFAST algorithm computes the $K$-sparse DFT of $x \in \mathbb{C}^N$,
- using $M = O(K)$ samples,
- in $O(K \log K)$ computations,
- with probability at least $1 - O(1/M)$. 

Noisy Setting: R-FFAST | Sample-Optimal Recovery with Near Linear Time
Noisy Setting: **R-FFAST** | *Sample-Optimal Recovery with Near Linear Time*

- $K = 20$ and $40$
- $N = 0.124$ million
- Sample cost $M = 2940$
Noisy Setting: **R-FFAST** | Near Sample-Optimal Recovery with Sub-linear Time

**Theorem:** R-FFAST algorithm computes the $K$-sparse DFT of $\mathbf{x} \in \mathbb{C}^N$,

- using $M$ noisy samples, where $M = O(K \log^{1.33} N)$,
- with probability at least $1 - O(1/M)$,
- in $O(K \log^{2.33} N)$ computations,
- for a finite and sufficiently high SNR.

### Noiseless vs. Noisy

**Matching 2 points of a noiseless sinusoid**

\[
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times2} & \cdots & W^{19\times19}
\end{bmatrix}
\]

**Matching $O(\log N)$ random points of a noisy sinusoid**

\[
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W & W^2 & \cdots & W^{19} \\
1 & W^2 & W^4 & \cdots & W^{2\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^k & W^{2k} & \cdots & W^{k\times19} \\
1 & W^{19} & W^{19\times2} & \cdots & W^{19\times19}
\end{bmatrix}
\]
Noisy Setting: **R-FFAST** | Near Sample-Optimal Recovery with Sub-linear Time

**Noisy**

Matching $O(\log N)$ random points of a noisy sinusoid

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & W & W^2 & \ldots & W^{19} \\
1 & W^2 & W^4 & \ldots & W^{2\times19} \\
1 & W^3 & W^6 & \ldots & W^{3\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^k & W^{2k} & \ldots & W^{k\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times2} & \ldots & W^{19\times19}
\end{bmatrix}
\]

**Noisy (sub-linear)**

Matching $O(\log N)$ random pieces of a noisy sinusoid

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 \\
1 & W & W^2 & \ldots & W^{19} \\
1 & W^2 & W^4 & \ldots & W^{2\times19} \\
1 & W^3 & W^6 & \ldots & W^{3\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^k & W^{2k} & \ldots & W^{k\times19} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W^{19} & W^{19\times2} & \ldots & W^{19\times19}
\end{bmatrix}
\]

Each piece is of length $O(\log^{0.3} N)$
Noisy Setting: **R-FFAST** | Near Sample-Optimal Recovery with Sub-linear Time

Noisy $O(\log N)$ random starts
Noisy Setting: **R-FFAST**  |  Near Sample-Optimal Recovery with Sub-linear Time

- For each random start, take consecutive delays spaced by $2^i$

$d_0 = 1$
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time

- For each random start, take consecutive delays spaced by $2^i$
- [1989’Kay] provides an unbiased and efficient estimate of $2^i \omega$

Noisy $O(\log N)$ random starts
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time

- For each random start, take consecutive delays spaced by $2^i$
- [1989'Kay] provides an unbiased and efficient estimate of $2^i \omega$
- $O(\log^{1/3} N)$ consecutive rows are sufficient for matching each piece
Noisy Setting: R-FFAST  |  Near Sample-Optimal Recovery with Sub-linear Time

- **Piece 1**: estimates $\omega$ with no ambiguity
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time

- **Piece 1**: estimates $\omega$ with no ambiguity
- **Piece 2**: estimates $2\omega \implies$ unwrapping leads to 2 ambiguities
Noisy Setting: **R-FFAST** | Near Sample-Optimal Recovery with Sub-linear Time

- **Piece 1**: estimates $\omega$ with no ambiguity
- **Piece 2**: estimates $2\omega \implies$ unwrapping leads to 2 ambiguities
- **Piece $i$**: estimates $2^i\omega \implies$ unwrapping leads to $2^i$ ambiguities
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time

- accuracy improves dyadically $2^i$
- each piece provides 1 information bit
- $\log N$ pieces $\implies N$ locations
Computing Sparse DFT | Some Concrete Examples of FFAST

Original Brain image

Fourier domain of the Brain Image

Brain image reconstructed by FFAST

• Brain image of dimension $N = 504 \times 504 \approx 0.25$ million.

• MRI: samples are taken in the Fourier domain.

• FFAST recovers the image using $60\%$ Fourier samples.
Computing Sparse DFT | Numerical Comparisons

**Sub-linear time performance:**
- Signal length increased 15 fold.
- Processing time less than 3 fold.

**Sample Complexity:**
- This verifies the sample overhead of fast search over the slow search.
Discussions:

- Sparse WHT
- Compressed Sensing
IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 41, NO. 3, MARCH 1993

The Poorman's Transform: Approximating the Fourier Transform without Multiplication

Michael P. Lamoureux

Abstract—A time domain to frequency domain transformation for sampled signals is described, which is computed with only additions and trivial complex multiplications. This Poorman’s transform is an approximation to the usual Fourier transform, obtained by quantizing the Fourier coefficients to the four values \([-1, 1, 1, 1]\), and is especially useful when multiplication is expensive. For the general case of an \(N\)-point quantization, an analytic formula is given for the error in the approximation, which involves only contributions from aliased harmonics. Continuous time signals are considered, where the approximation is exact for band-limited signals.
Walsh-Hadamard Transform |

- **Noiseless results** [2013’Scheibler *et al*]
  Scheibler, R., Haghighatshoar, S., Vetterli, M., 2013 51st Annual Allerton Conf. on Communication, Control, and Computing (pp. 1250-1257).
Walsh-Hadamard Transform

- **Noiseless results** [2013’Scheibler et al]
  Scheibler, R., Haghighatshoar, S., Vetterli, M., 2013 51st Annual Allerton Conf. on Communication, Control, and Computing (pp. 1250-1257).

- **Noisy results** [2014’Li et al]
Conclusion

• FFAST algorithm for computing $K$-sparse DFTs
  
  – exploits coding-theory principle (i.e., sparse-graph alias codes)
  
  – **Noiseless:**
    
    * $M = O(K)$ samples and $T = O(K \log K)$ run-time
  
  – **Noisy:**
    
    * sample-optimal recovery $M = O(K \log N)$
      with near-linear run-time $T = O(N \log N)$
    
    * near sample-optimal recovery $M = O(K \log^{1.3} N)$
      with sub-linear run-time $T = O(K \log^{2.3} N)$

• Extensions to sparse WHT and compressed sensing
  
  – **Sparse WHT**
    
    https://www.eecs.berkeley.edu/~kannanr/assets/project_fft/WHT_noisy.pdf
  
  – **Compressed sensing using sparse-graph codes**
    
    http://www.eecs.berkeley.edu/~xiaoli/FR_CS_SGC.pdf
Future Directions

• More general sparsity and signal models

• Off-grid frequency estimation

• Practical applications in
  – MRI
  – Optical imaging such as Fourier Ptychography
  – Phase retrieval $\mapsto$ “PhaseCode” design