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

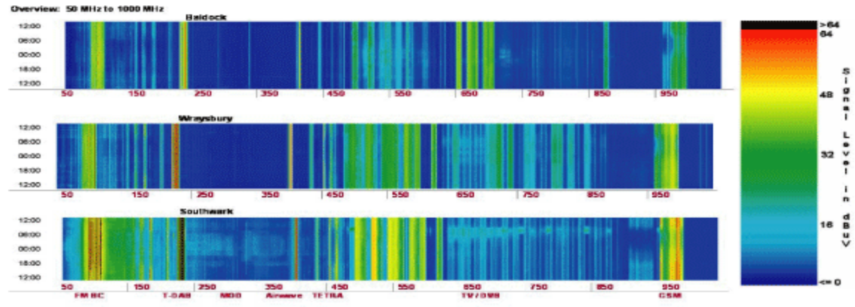
Acknowledgements

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

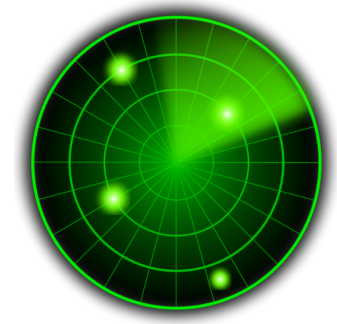
Acknowledgements

Piotr Indyk and **Dina Katabi**

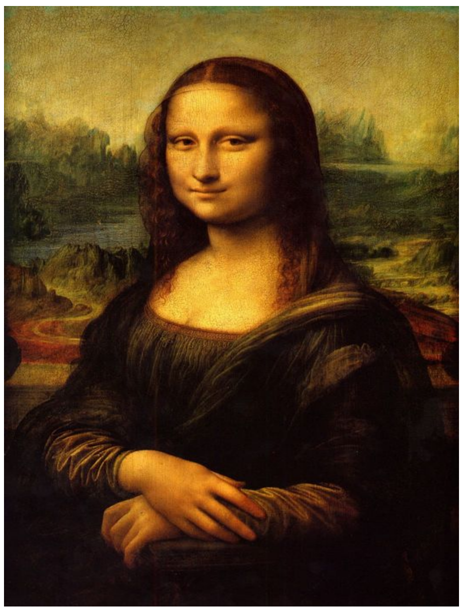
MRI



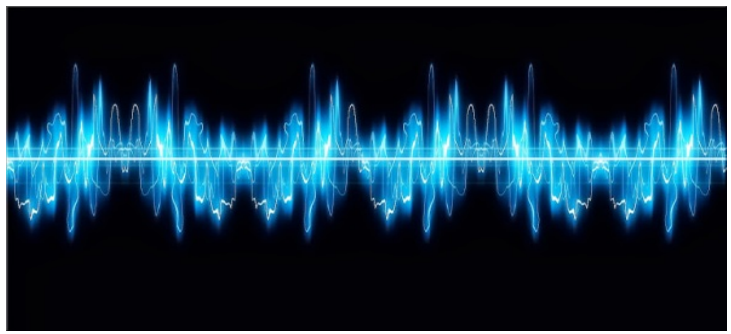
Cognitive Radio



Radar



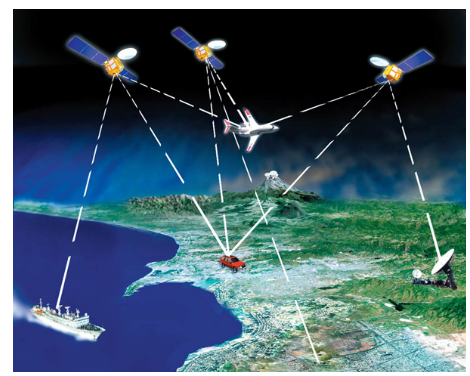
Images



Speech & audio

Ultrasound

Astronomy



GPS

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

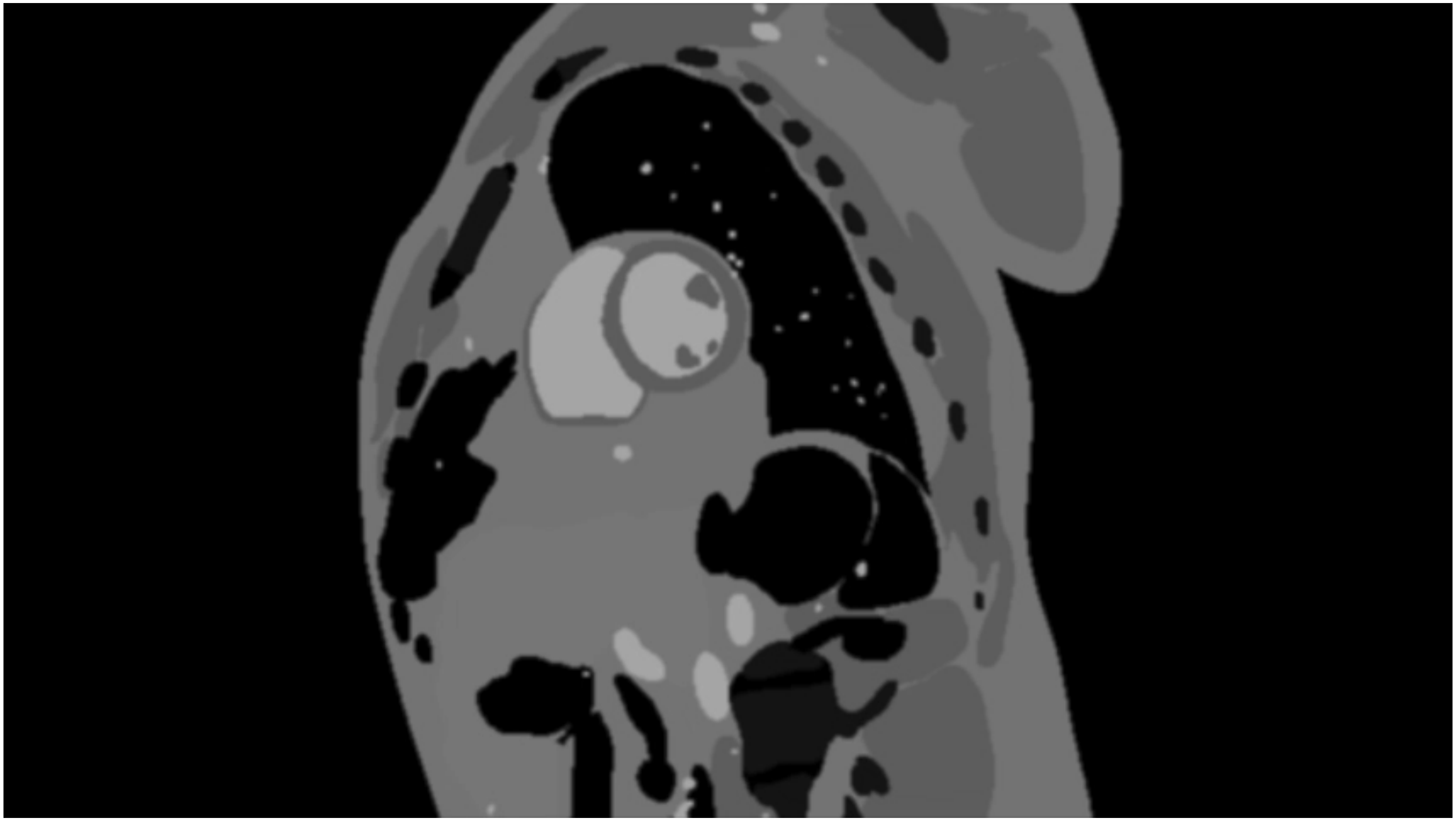
PART I:
Noiseless Recovery

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

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

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

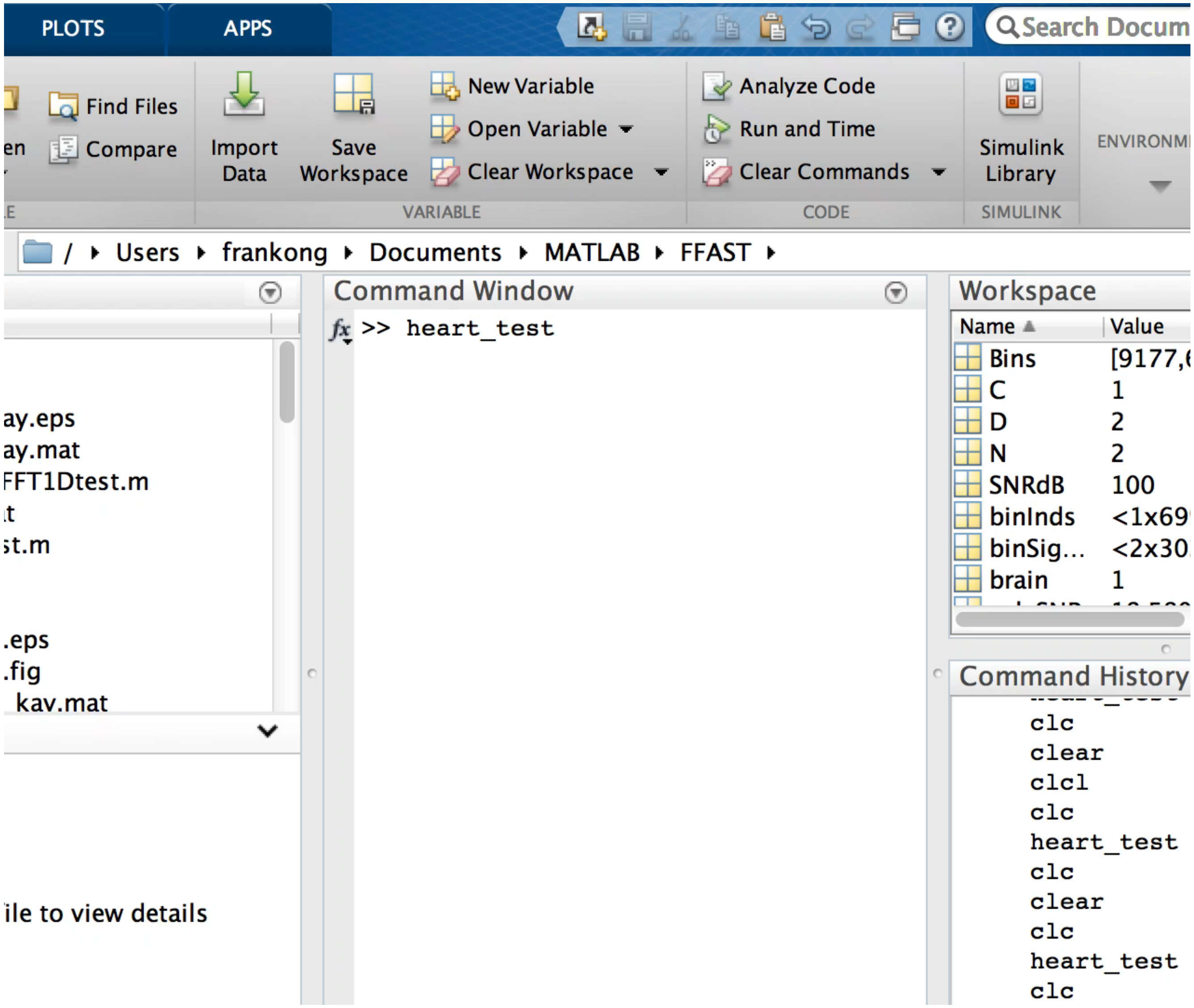
- Classical solution: FFT algorithm
  - Sample cost $= N$
  - Computational cost $= \mathcal{O}(N \log N)$
- With sparsity, what are the fundamental bounds for **support recovery**?
  - Sample cost?
  - Computational cost?



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


PLOTS
APPS
Search Docum
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
ENVIRONM
VARIABLE
CODE
SIMULINK
/ Users frankong Documents MATLAB FFAST
Command Window
>> heart_test
Workspace
Name Value
Bins [9177,
C 1
D 2
N 2
SNRdB 100
binInds <1x69
binSig... <2x30
brain 1
ay.eps
ay.mat
FFT1Dtest.m
st.m
.eps
.fig
kav.mat
ile to view details
Command History
clc
clear
clcl
clc
heart_test
clc
clear
clc
heart_test
clc

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

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

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

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.

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

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

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

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

- 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

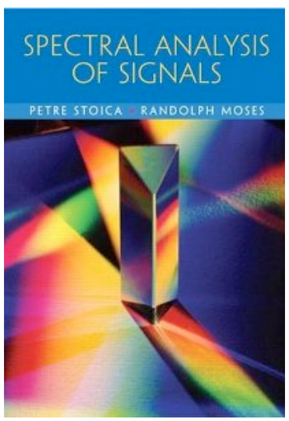
- Frequency estimation
  - Prony [1795]
  - Pisarenko ['73]
  - Vetterli et al. ['02]
  - more ···
- Subspace methods ['86]
  - MUSIC, ESPRIT etc.

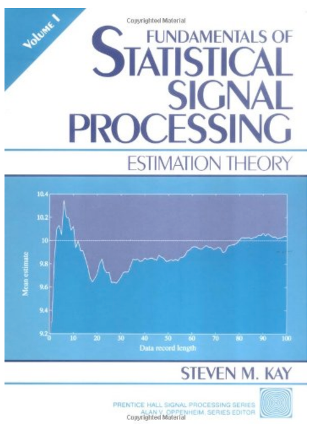
Compressed Sensing

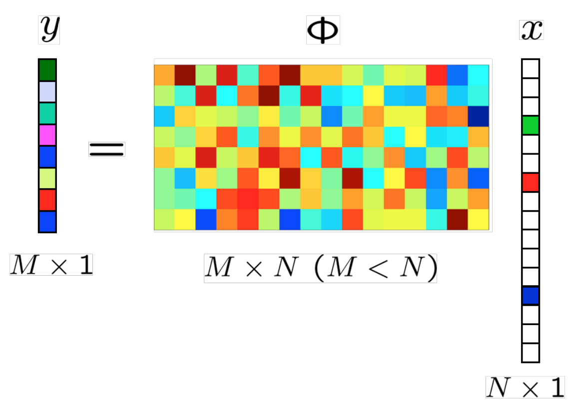
- Feng and Bresler [1996]
- Candes et. al [2006]
- Rauhut [2008]
- Wainwright [2009]
- Tang et. al [2012]
- more ···

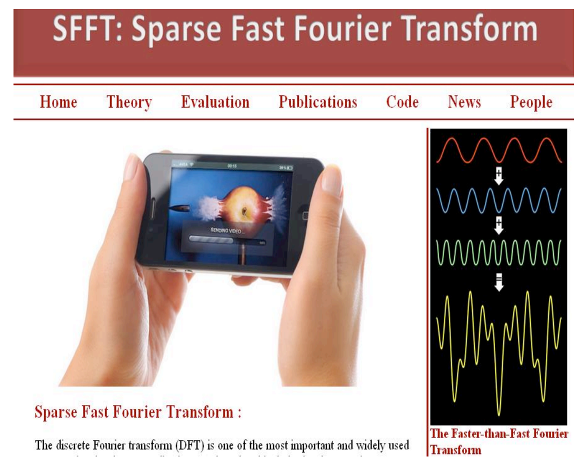
Sparse DFT

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





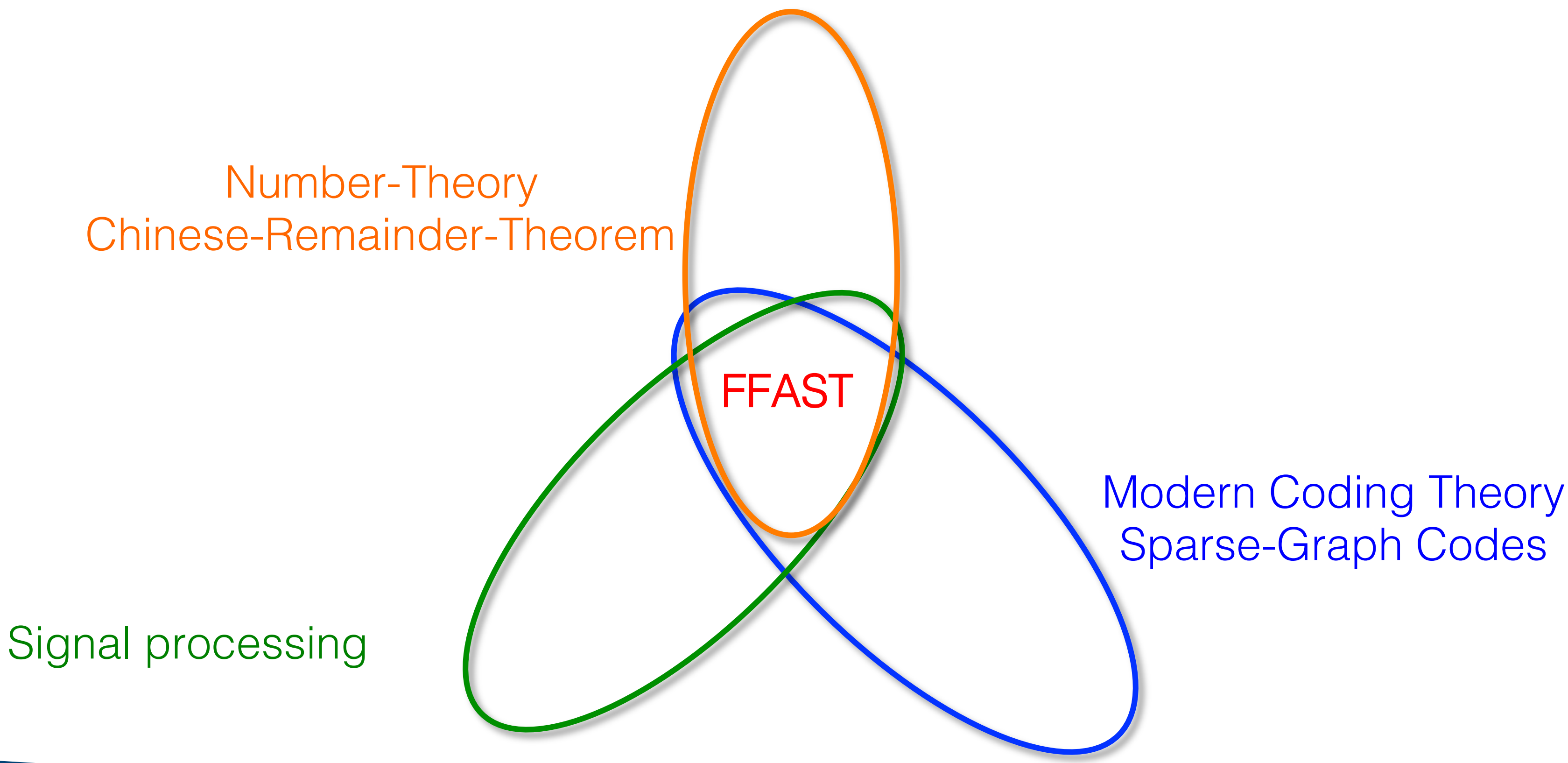




- More recent work on computing sparse DFT
  - http://groups.csail.mit.edu/netmit/sFFT/paper.html
- Recent advances in compressed sensing and sketching methods
- Our method:
  - targets support recovery instead of ℓ_2/ℓ_1 approximation
  - uses the design and analysis of sparse-graph codes
  - leverages harmonic retrieval methods in statistical signal processing

Computing Sparse DFT | Insights

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



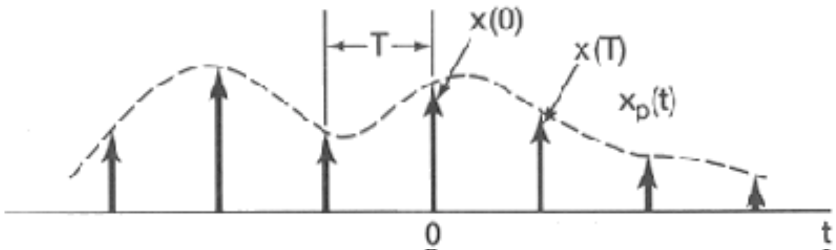
Computing Sparse DFT | Insights

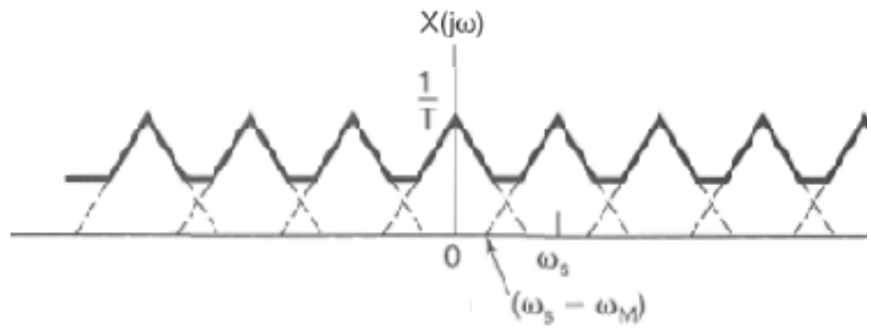
sub-sampling
below Nyquist rate → Aliasing in the frequency domain

clever sub-sampling
(for **sparse** case) → good "alias code"?

Chinese-Remainder-Theorem guided subsampling

Sparse graph codes





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

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

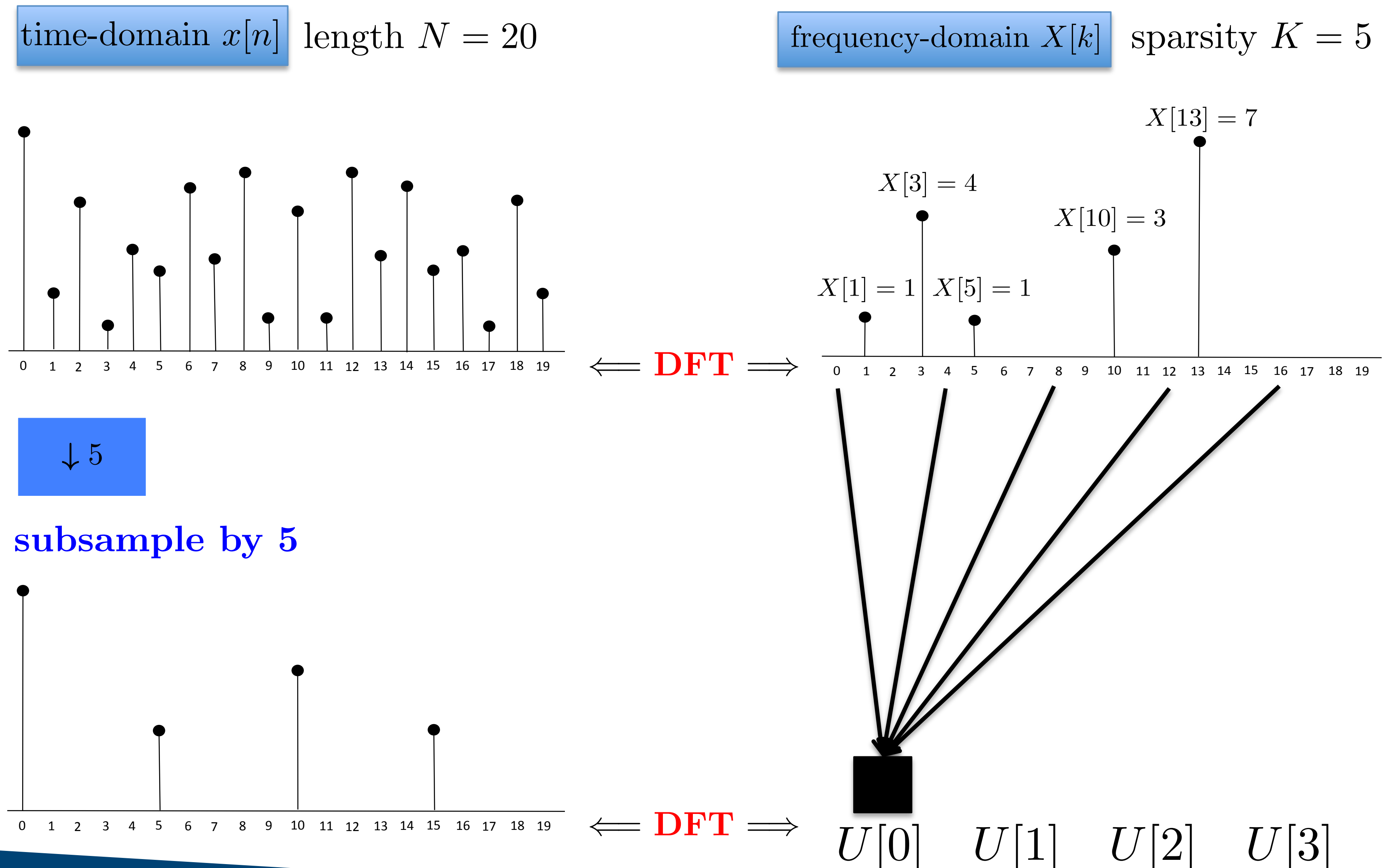
Chinese-Remainder-Theorem guided subsampling

Sparse graph codes

Coding-theoretic tools

- Design:
  - Randomized constructions of good sparse-graph codes.
- Analysis:
  - Density evolution
  - Martingale
  - Expander graph theory

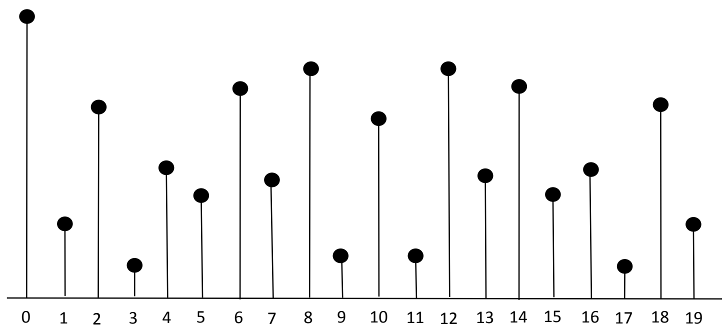
Computing Sparse DFT | Main Idea



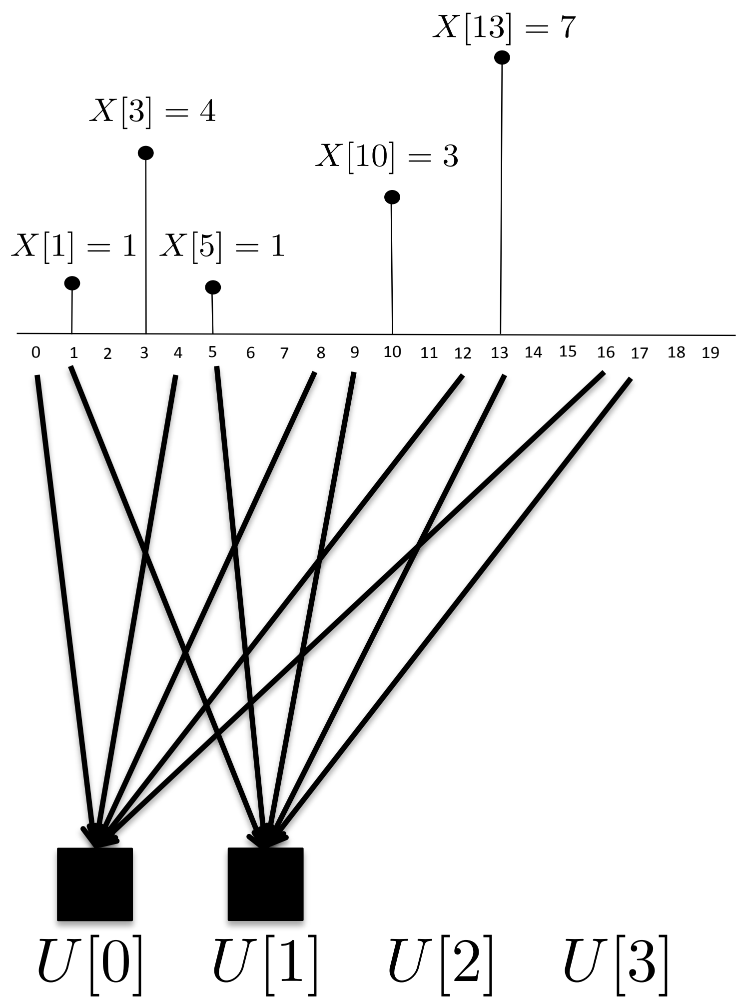
Computing Sparse DFT | Main Idea

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

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

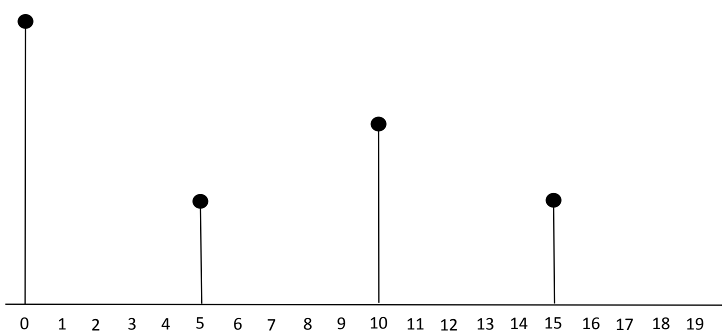


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

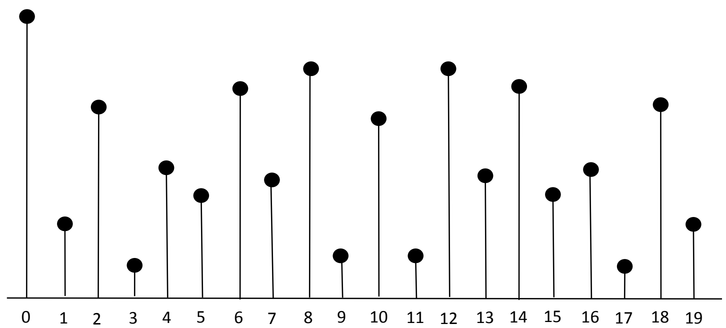


$\Longleftarrow$ **DFT** $\Longrightarrow$

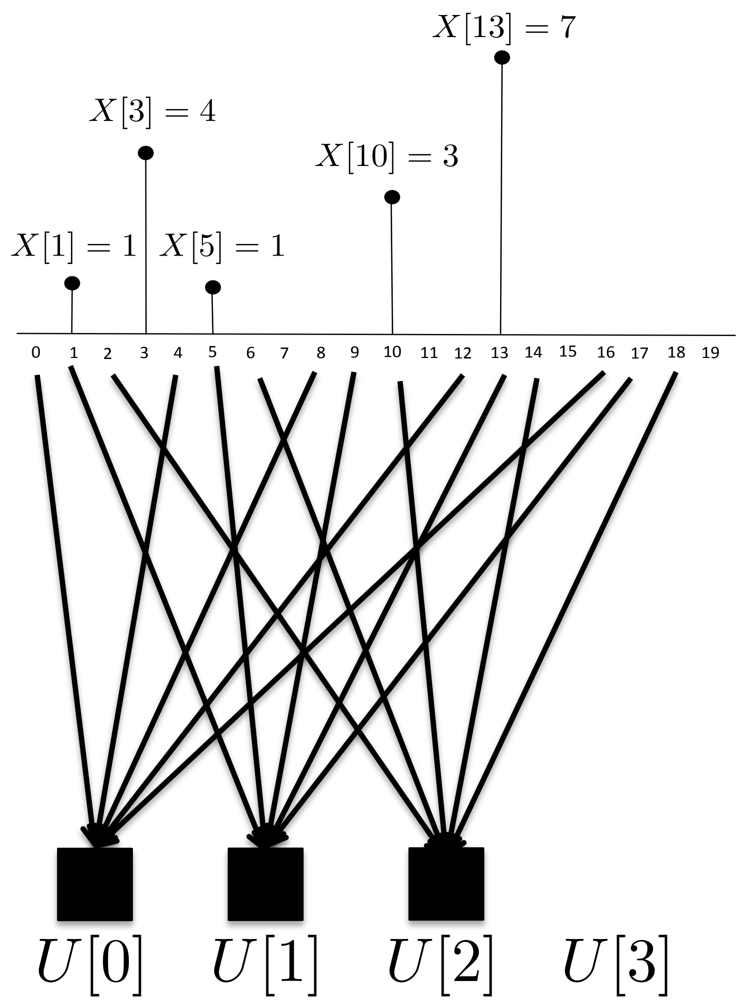
Computing Sparse DFT | Main Idea

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

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

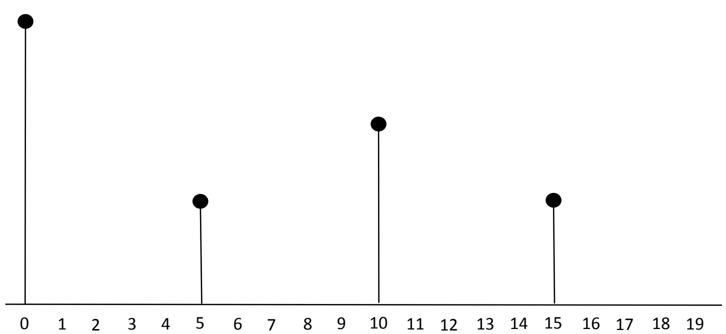


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

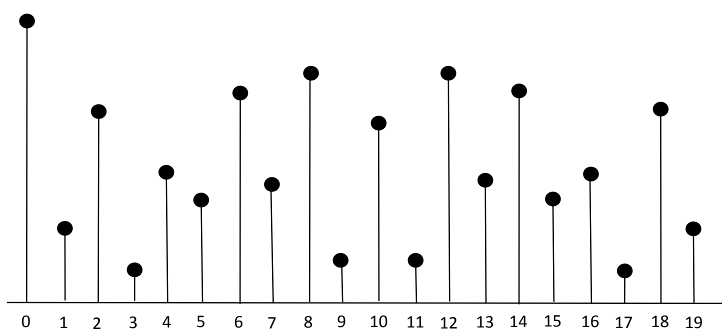


$\Longleftarrow$ **DFT** $\Longrightarrow$

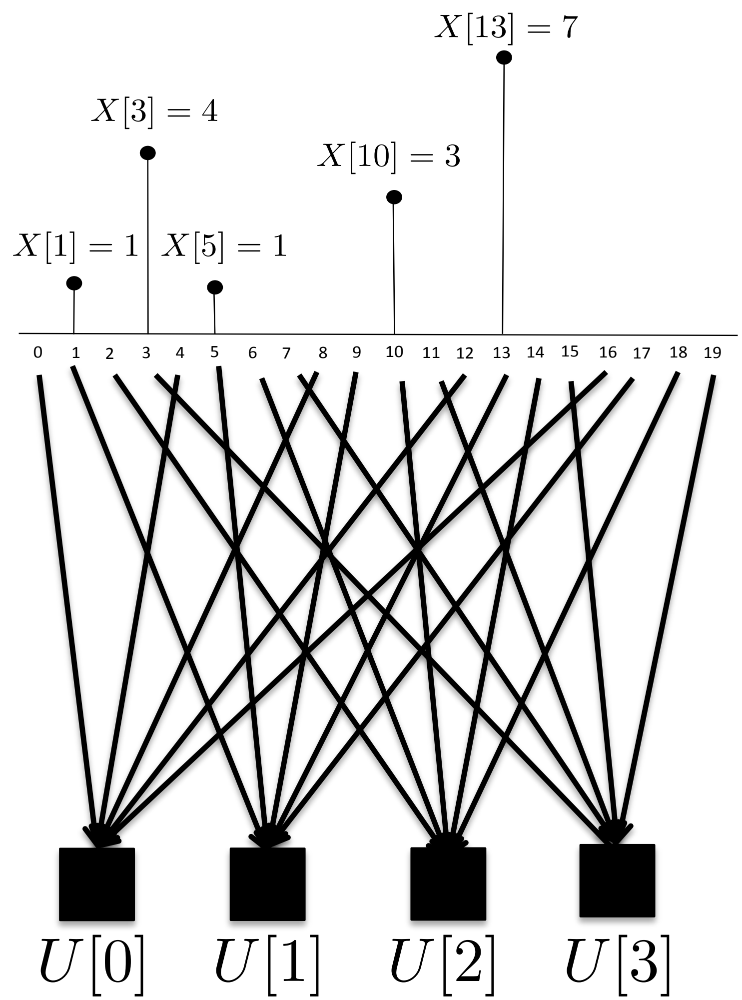
Computing Sparse DFT | Main Idea

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

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

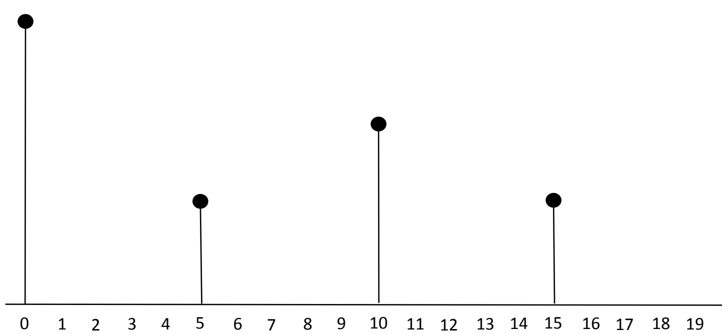


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

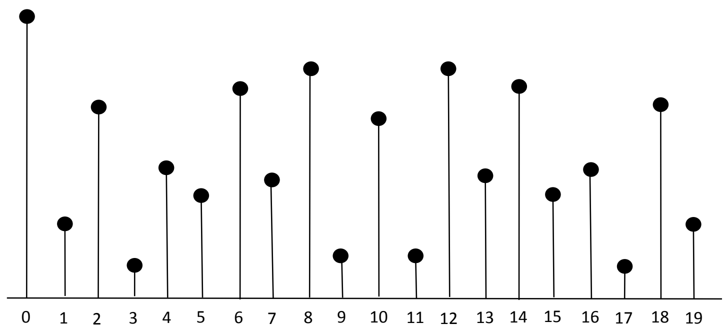


$\Longleftarrow$ **DFT** $\Longrightarrow$

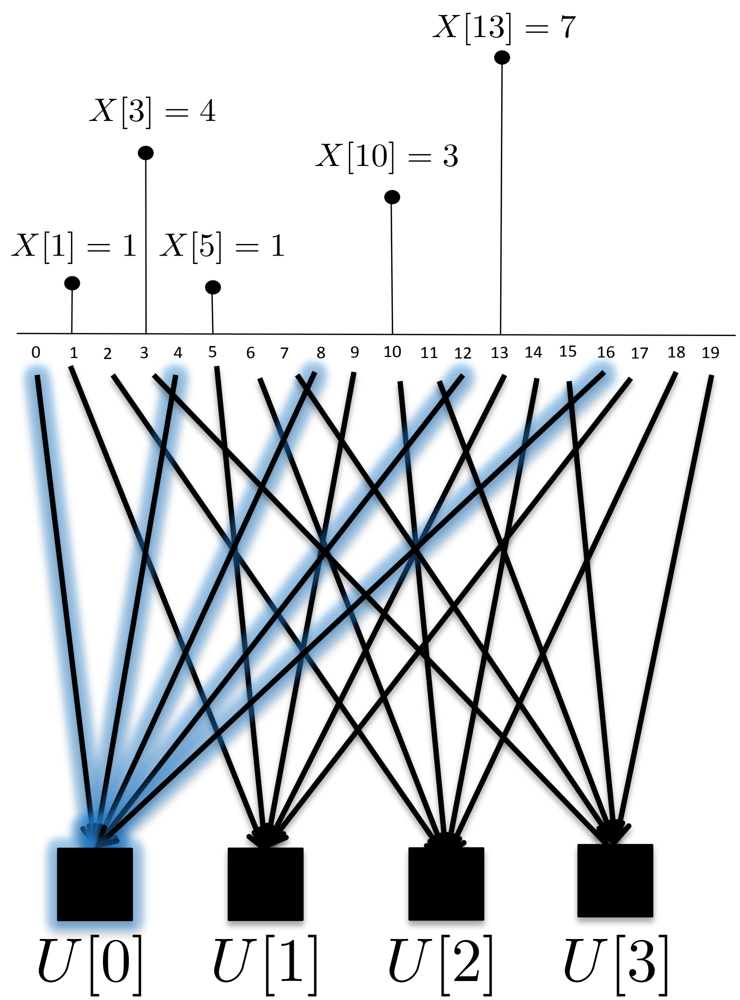
Computing Sparse DFT | Main Idea

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

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

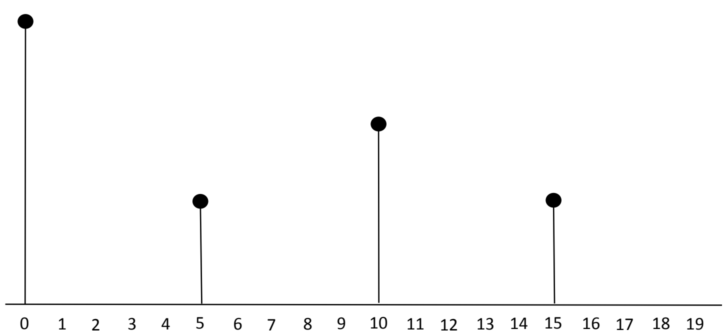


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

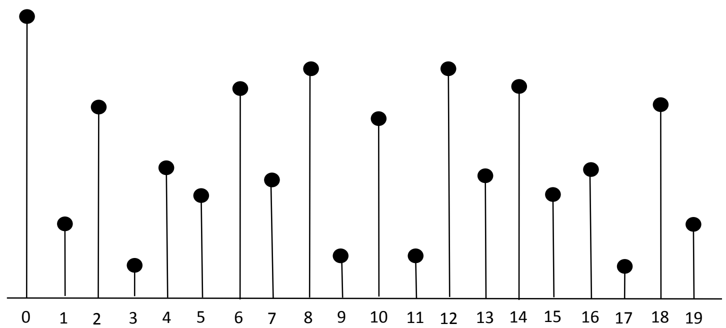


$\Longleftarrow$ **DFT** $\Longrightarrow$

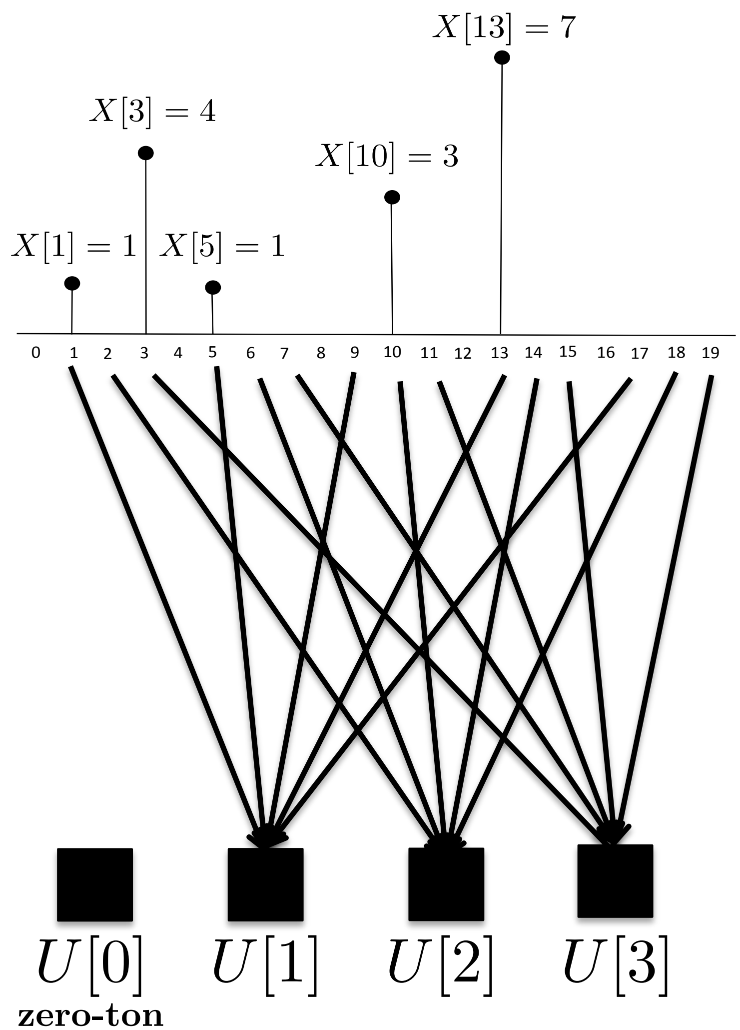
Computing Sparse DFT | Main Idea

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

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

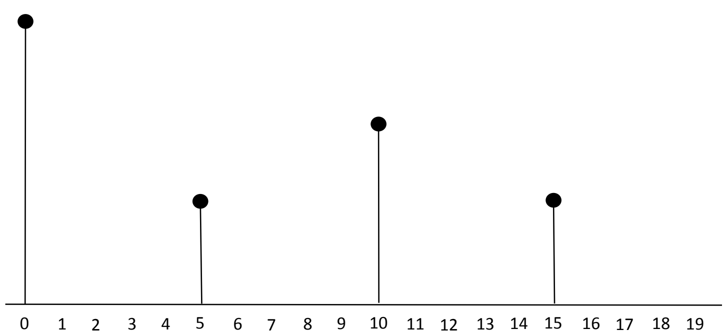


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

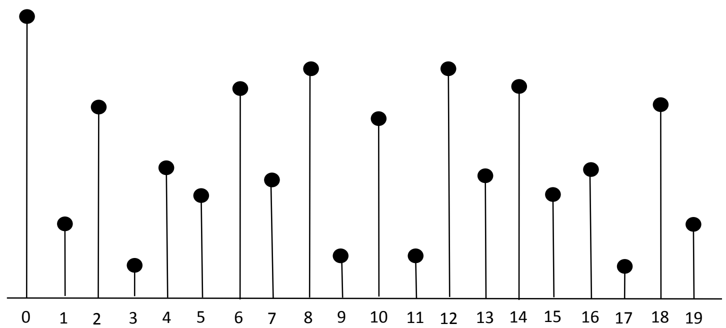


$\Longleftarrow$ **DFT** $\Longrightarrow$

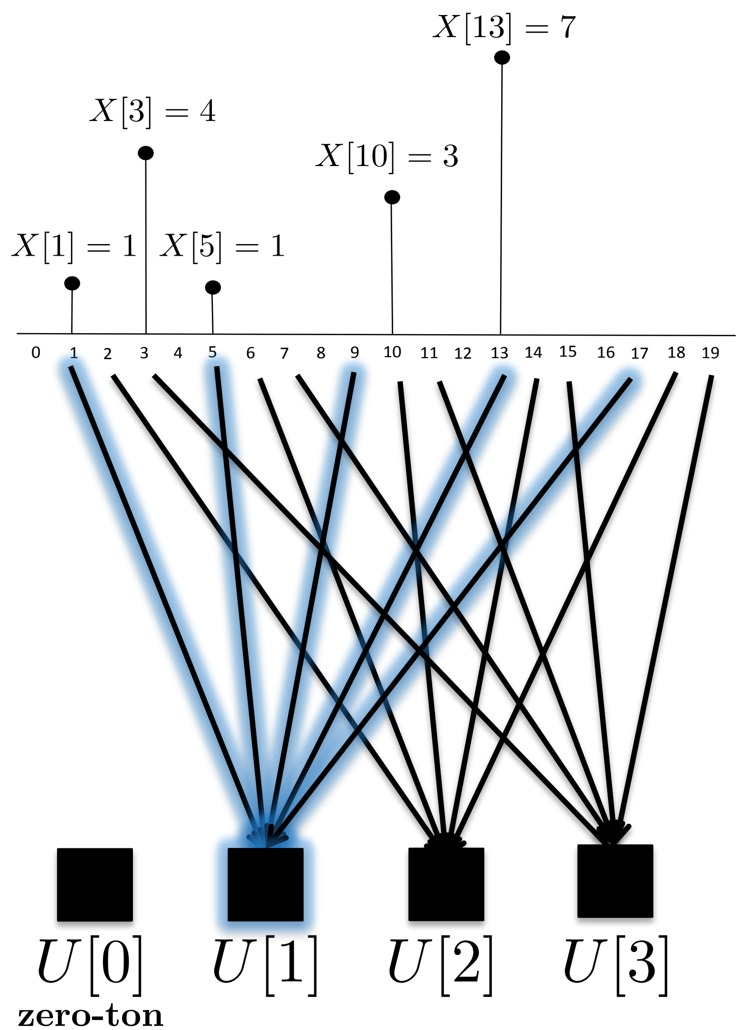
Computing Sparse DFT | Main Idea

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

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

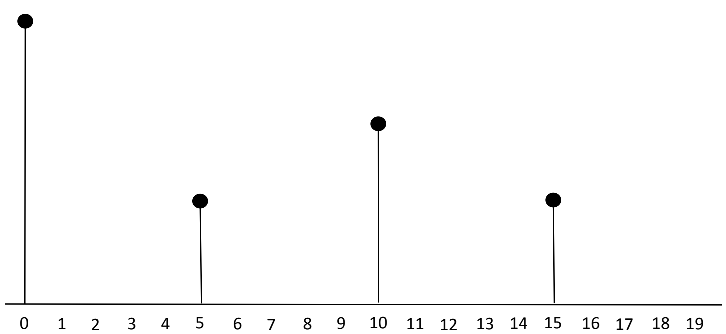


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

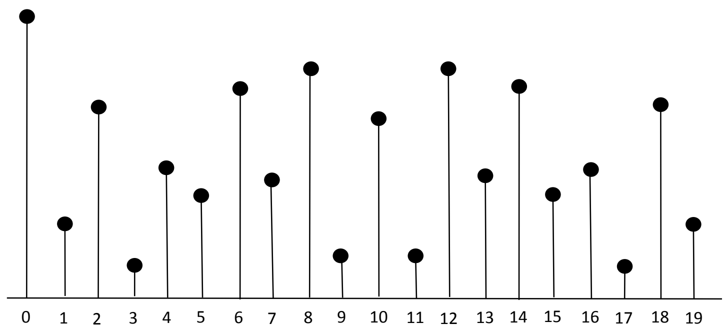


$\Longleftarrow$ **DFT** $\Longrightarrow$

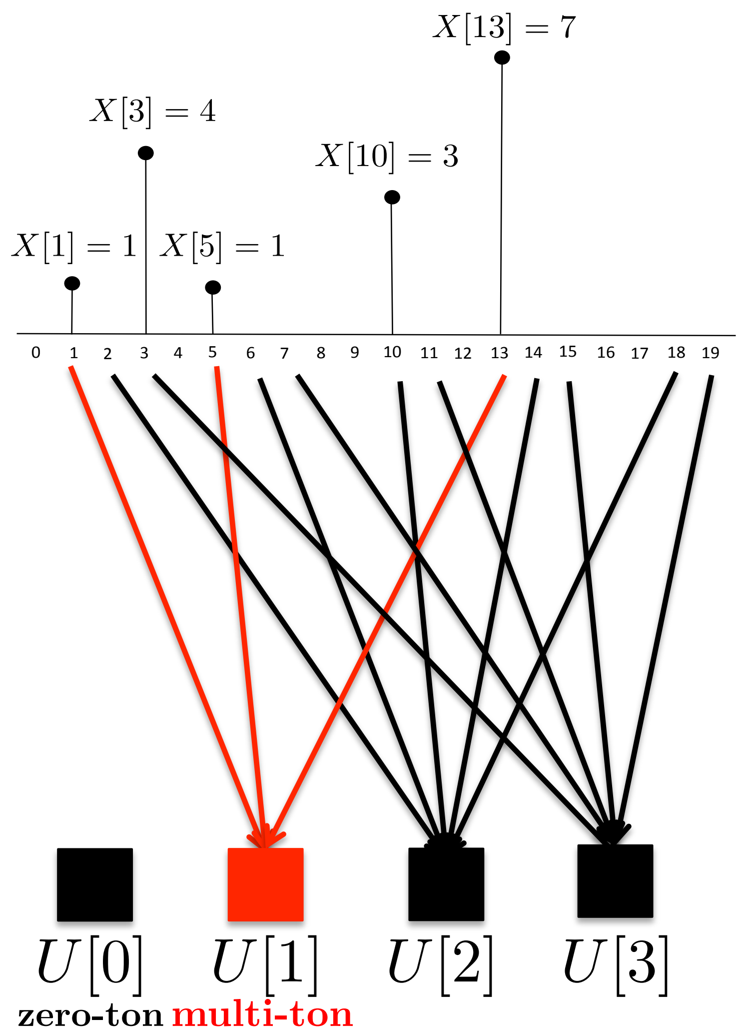
Computing Sparse DFT | Main Idea

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

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

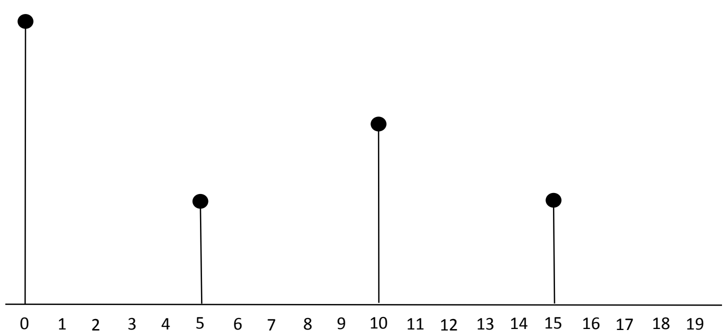


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

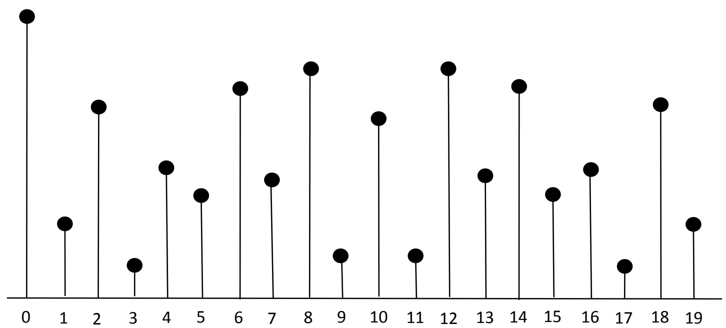


$\Longleftarrow$ **DFT** $\Longrightarrow$

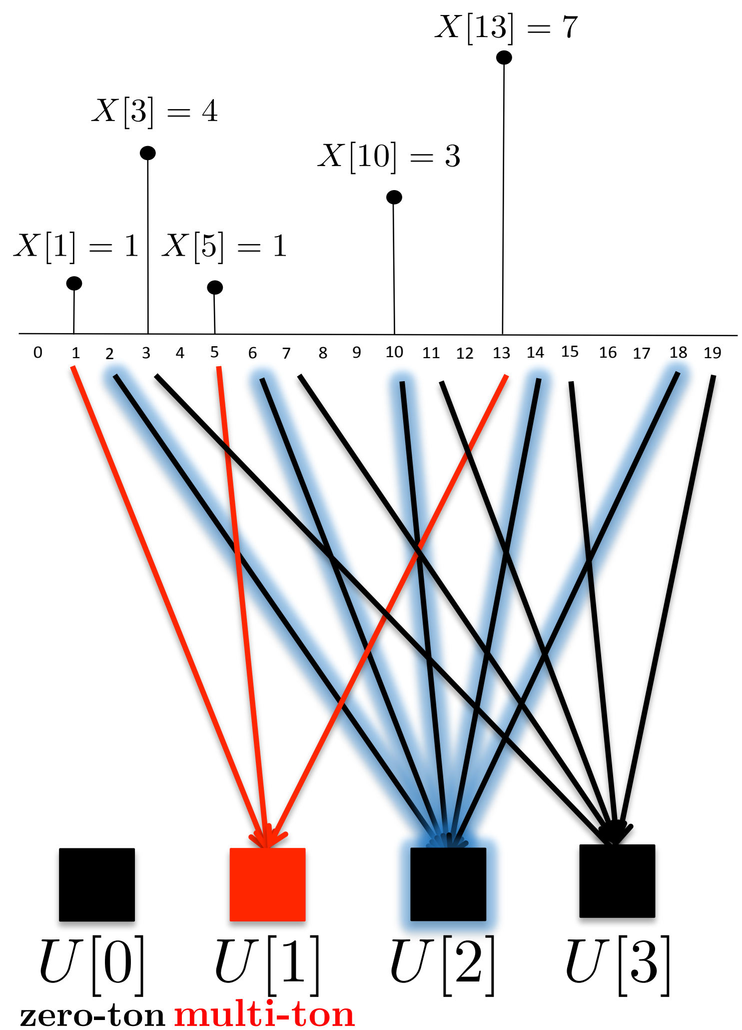
Computing Sparse DFT | Main Idea

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

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

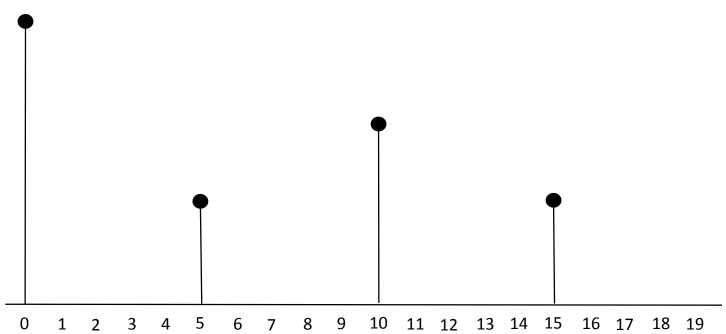


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

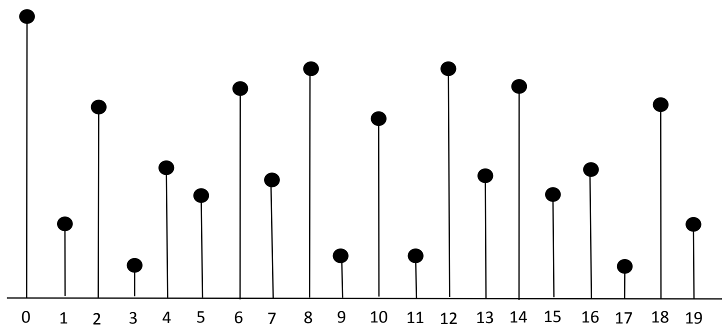


$\Longleftarrow$ **DFT** $\Longrightarrow$

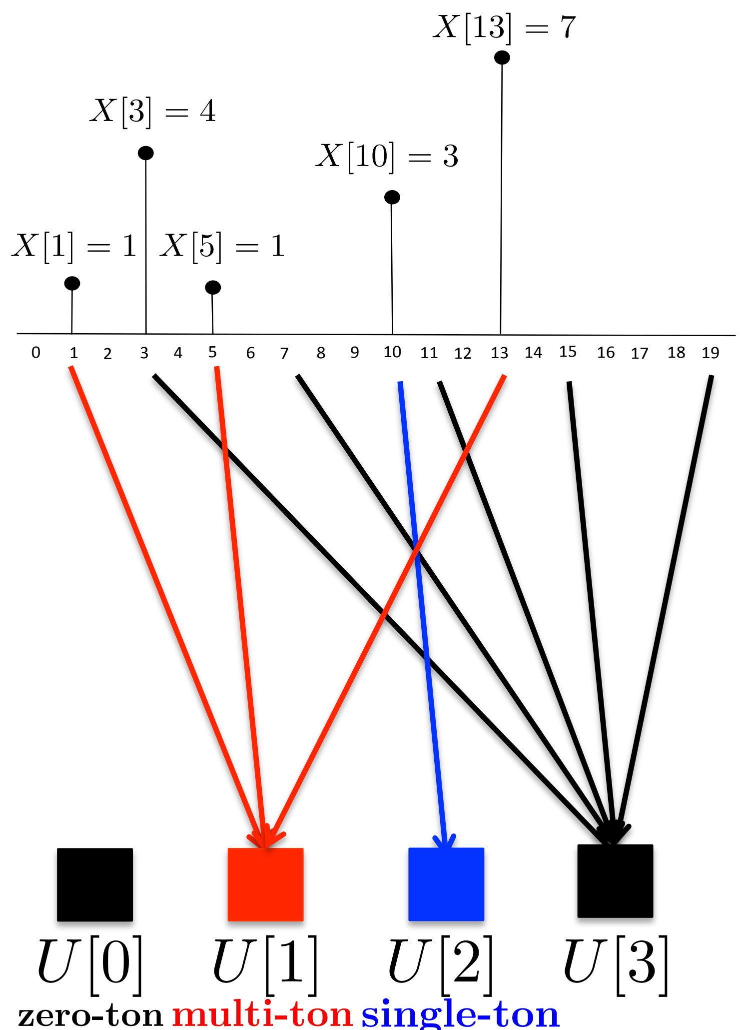
Computing Sparse DFT | Main Idea

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

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

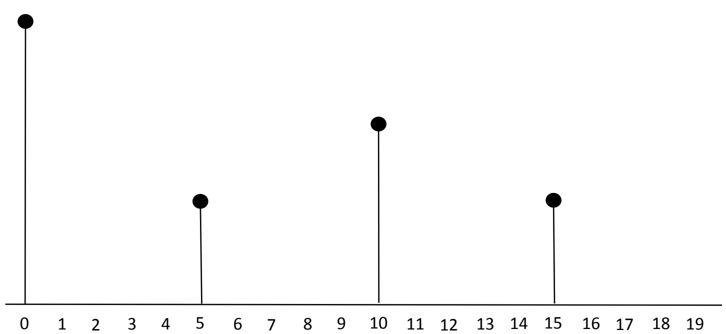


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

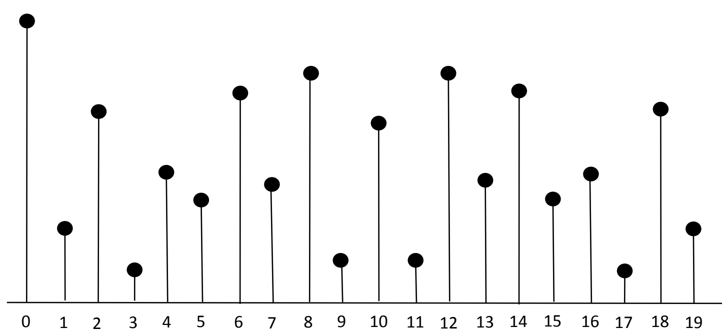


$\Longleftarrow$ **DFT** $\Longrightarrow$

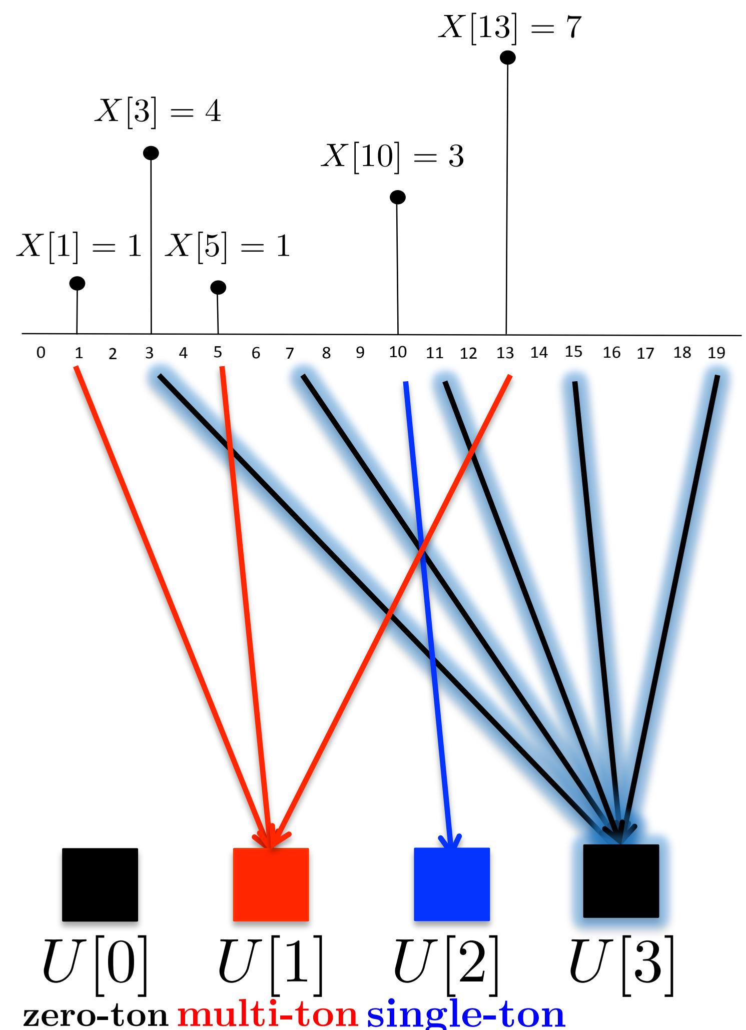
Computing Sparse DFT | Main Idea

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

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

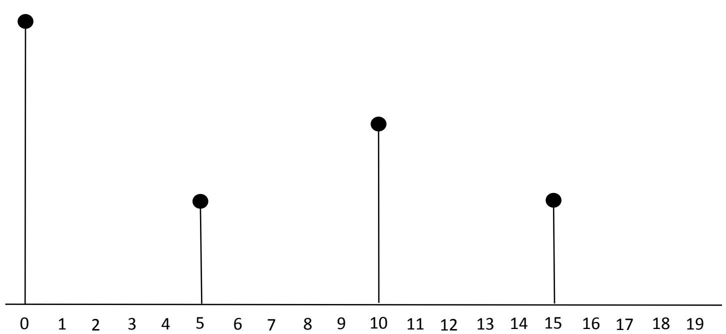


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

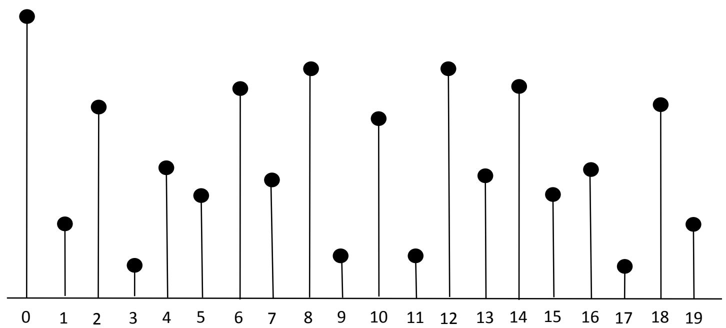


$\Longleftarrow$ **DFT** $\Longrightarrow$

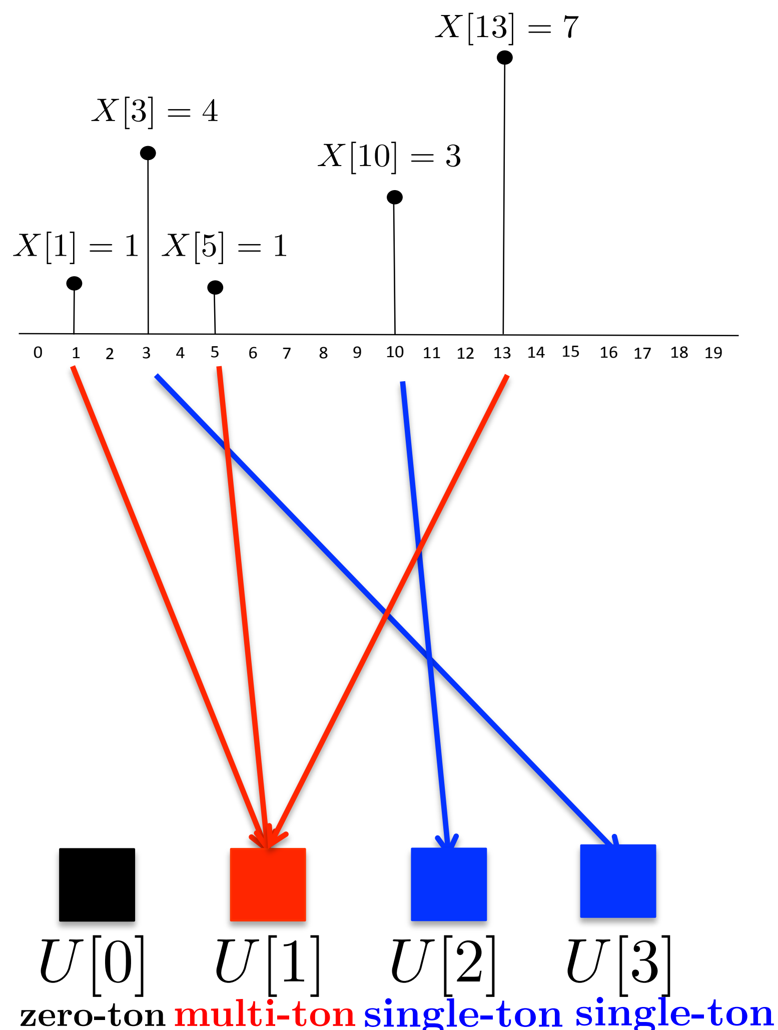
Computing Sparse DFT | Main Idea

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

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

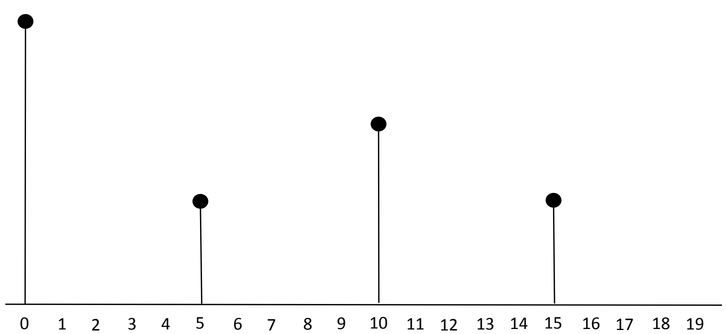


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

subsample by 5

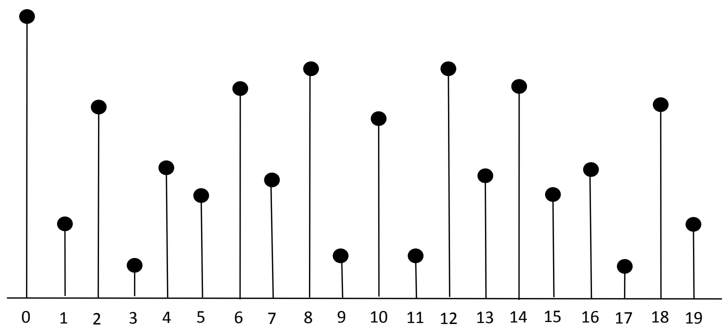


$\Longleftarrow$ **DFT** $\Longrightarrow$

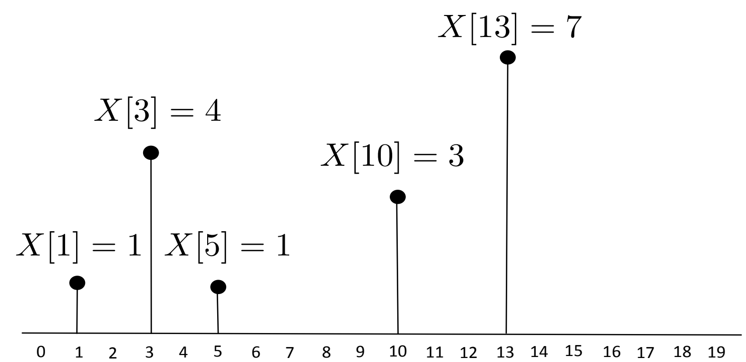
Computing Sparse DFT | Main Idea

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

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

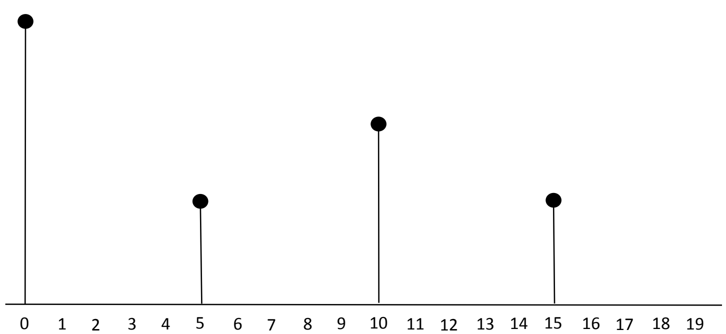


$\Longleftarrow$ **DFT** $\Longrightarrow$

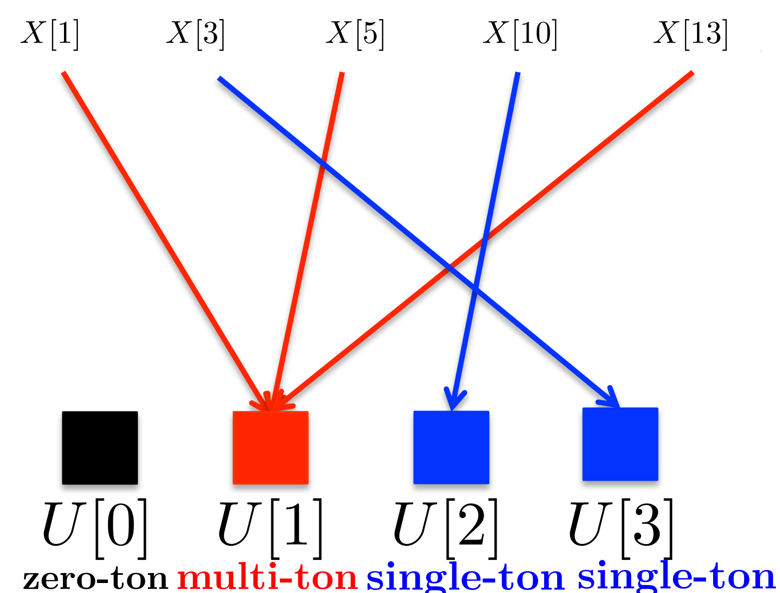


$\downarrow 5$

subsample by 5



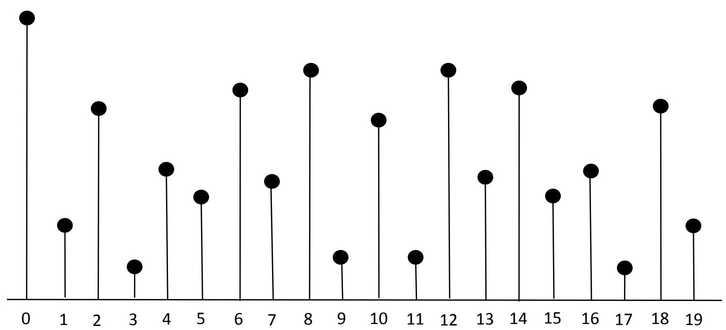
$\Longleftarrow$ **DFT** $\Longrightarrow$



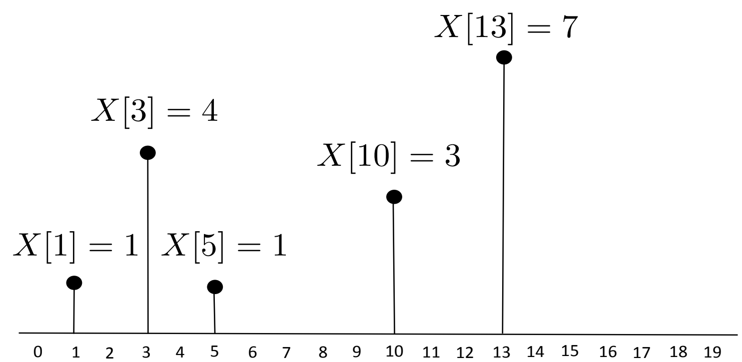
Computing Sparse DFT | Main Idea

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

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

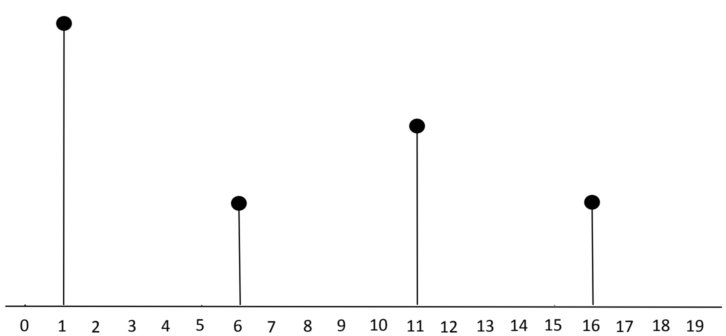


$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

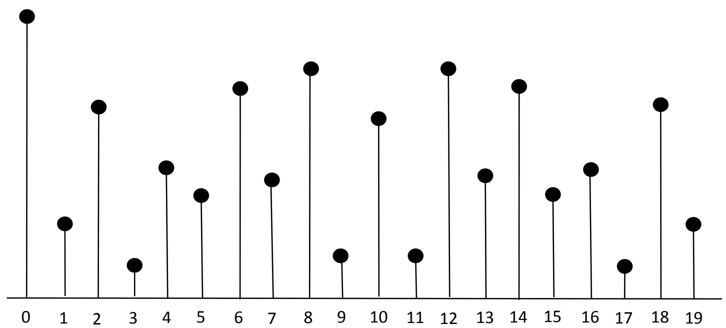
shift & subsample by 5



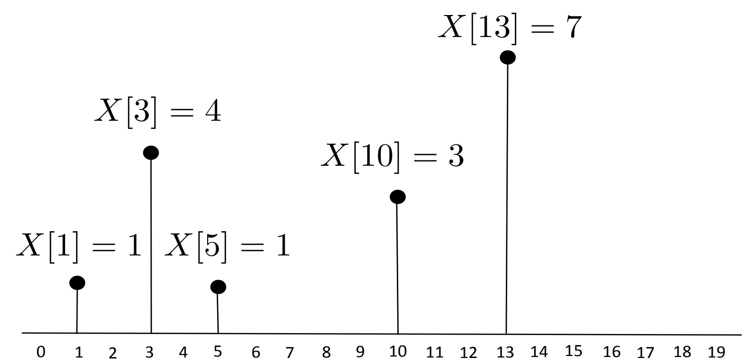
Computing Sparse DFT | Main Idea

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

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



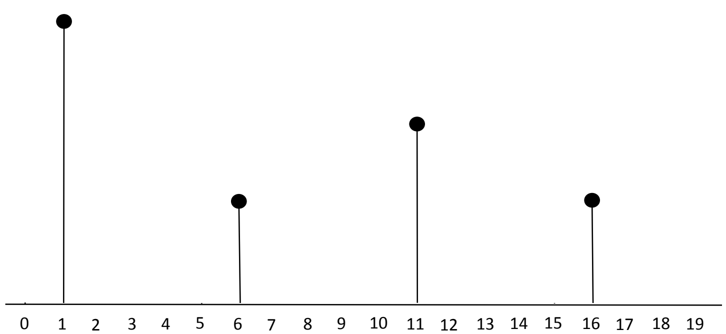
$\Longleftarrow$ **DFT** $\Longrightarrow$



$\downarrow 5$

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

shift & subsample by 5



$\Longleftarrow$ **DFT** $\Longrightarrow$

subscript U_S suggests **shift**

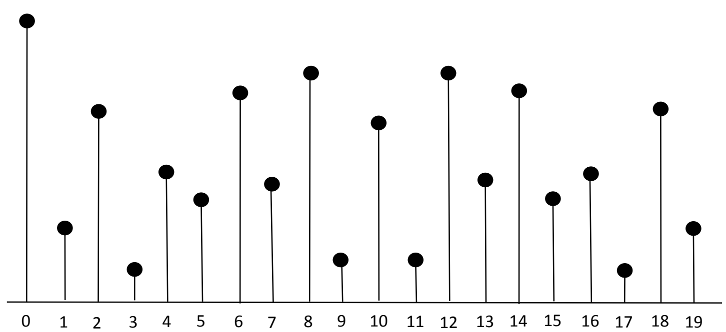
Our Measurements

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

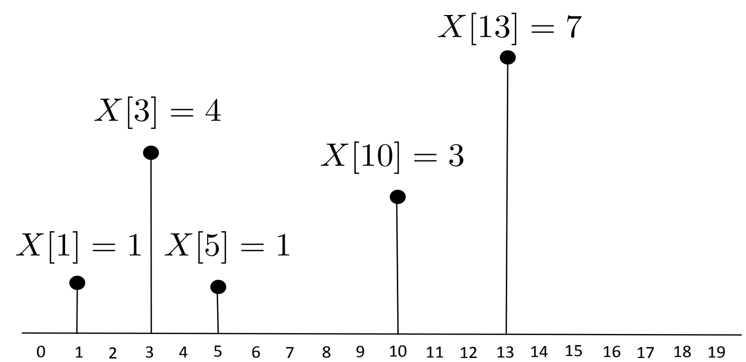
Computing Sparse DFT | Main Idea

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

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

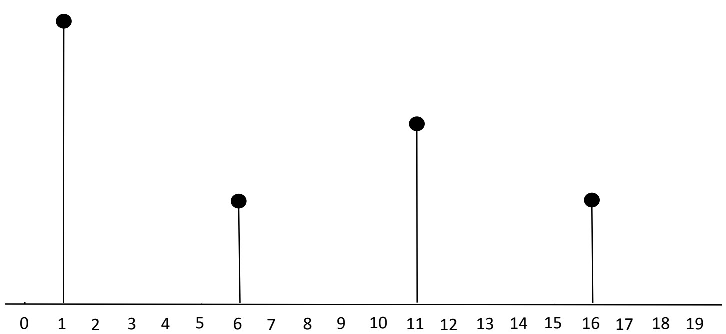


$\Longleftarrow$ **DFT** $\Longrightarrow$

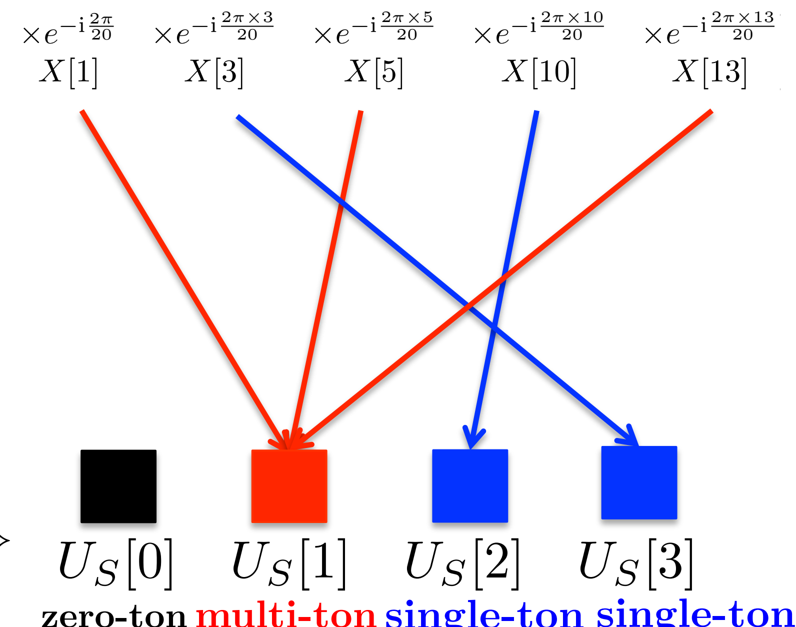


$\downarrow 5$

shift & subsample by 5

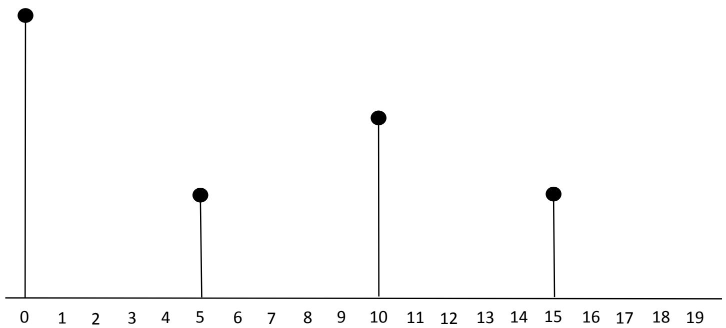


$\Longleftarrow$ **DFT** $\Longrightarrow$

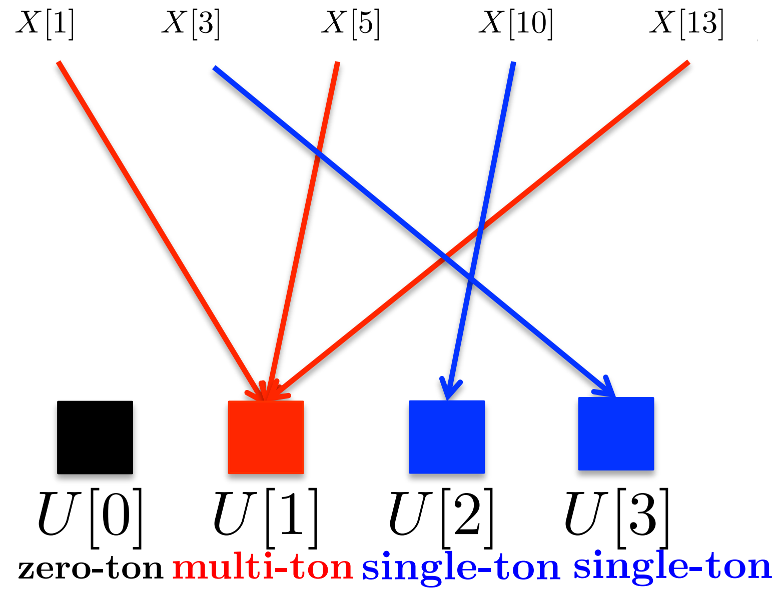


Computing Sparse DFT | Main Idea

subsample by 5

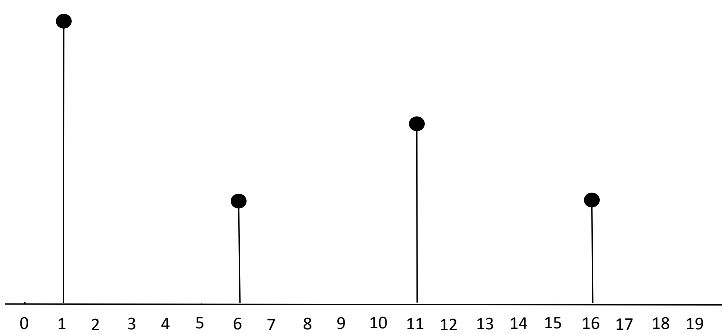


$\Longleftarrow$ **DFT** $\Longrightarrow$

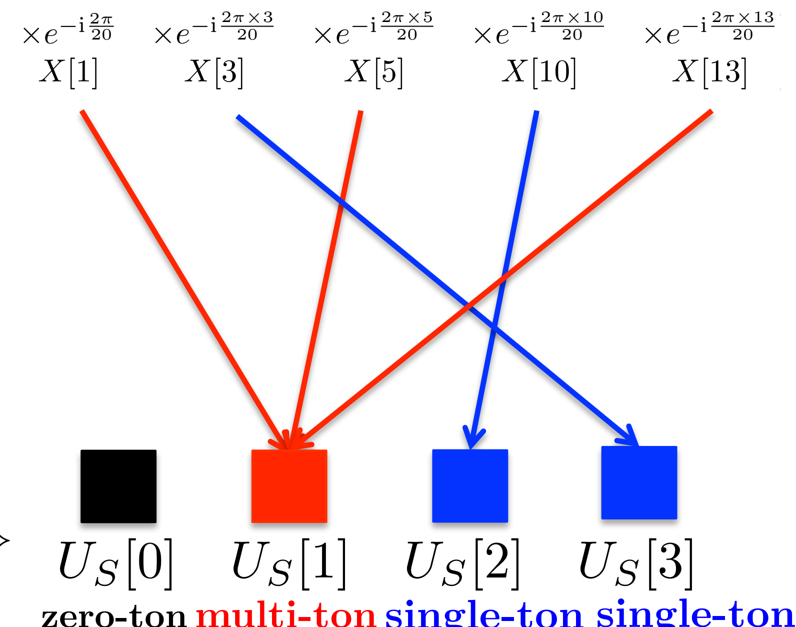


$\downarrow 5$

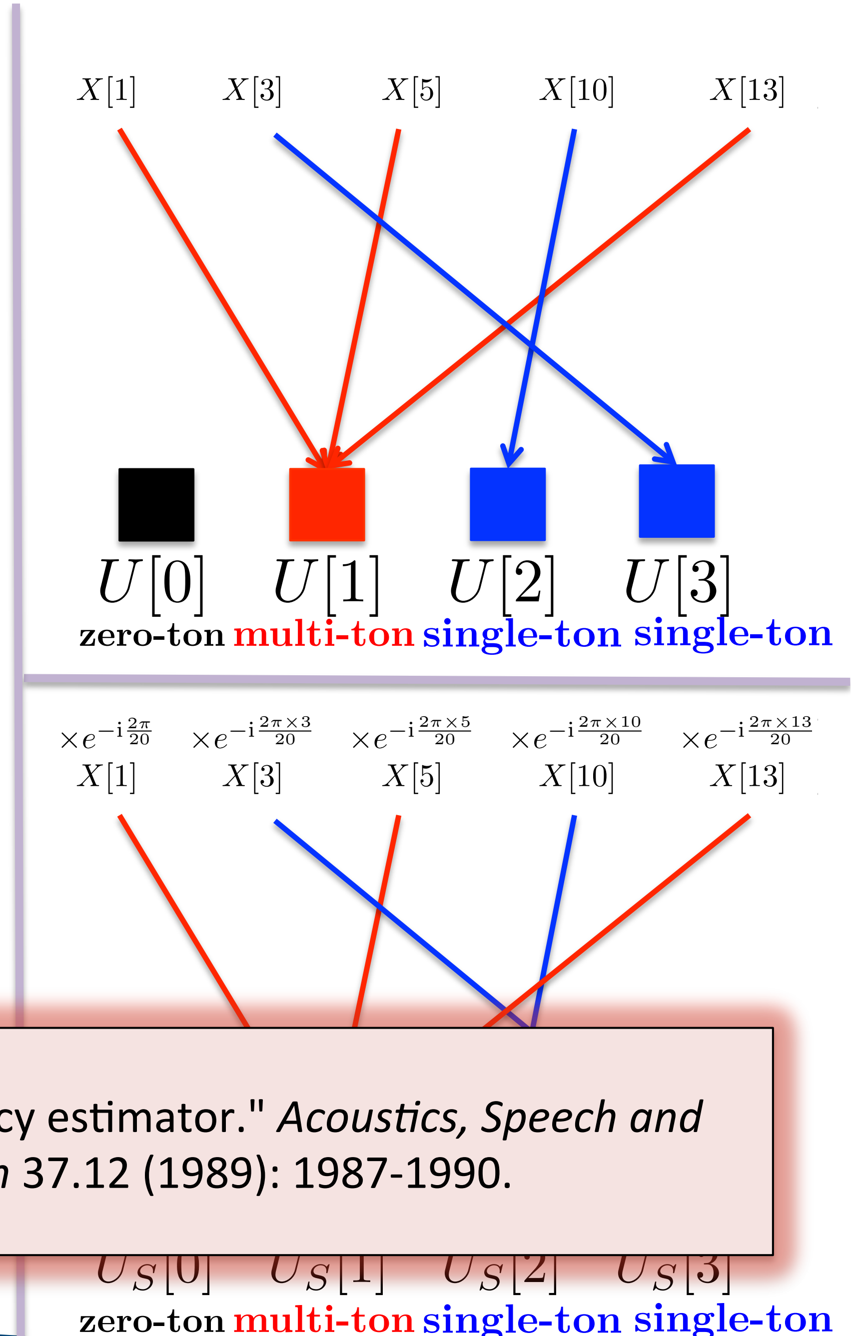
shift & subsample by 5



$\Longleftarrow$ **DFT** $\Longrightarrow$

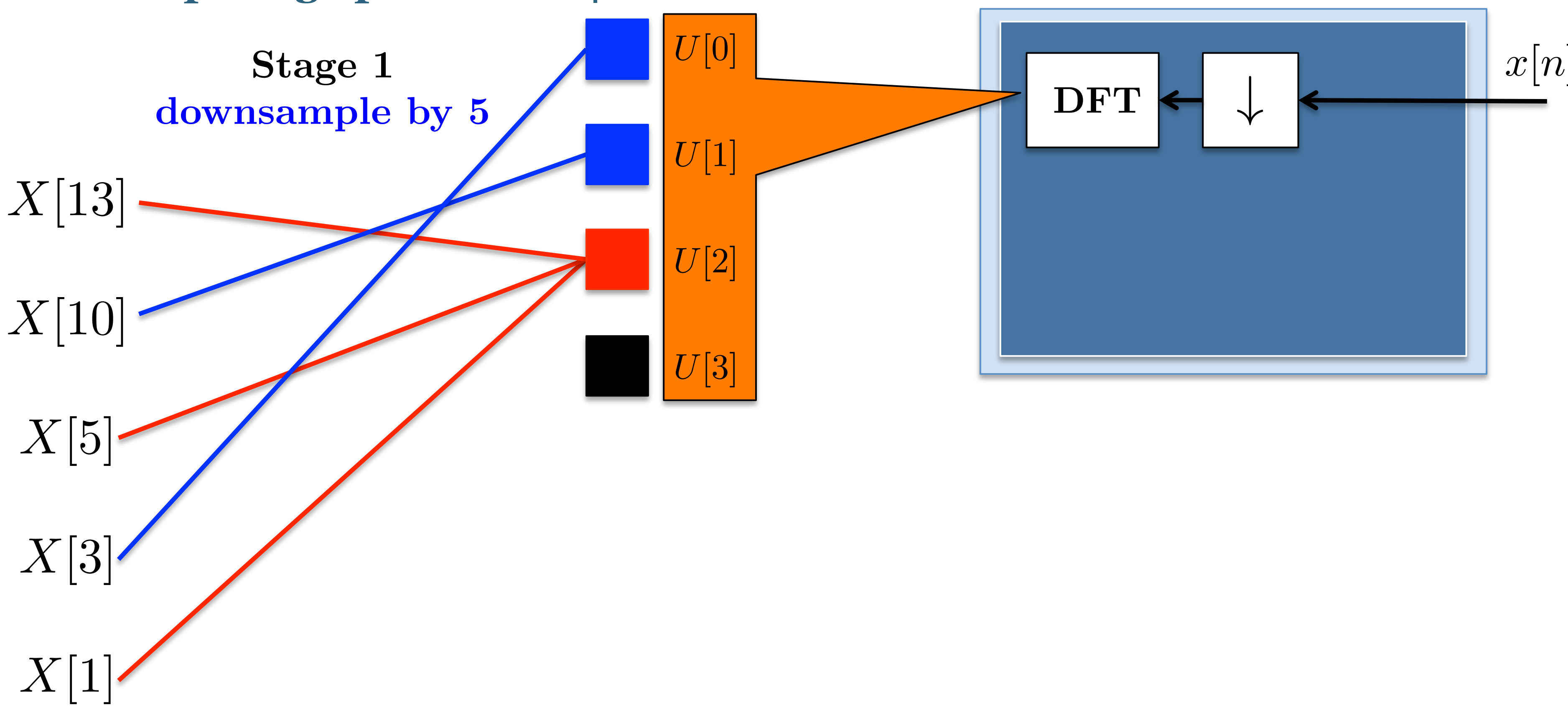


Why do we need shifts?

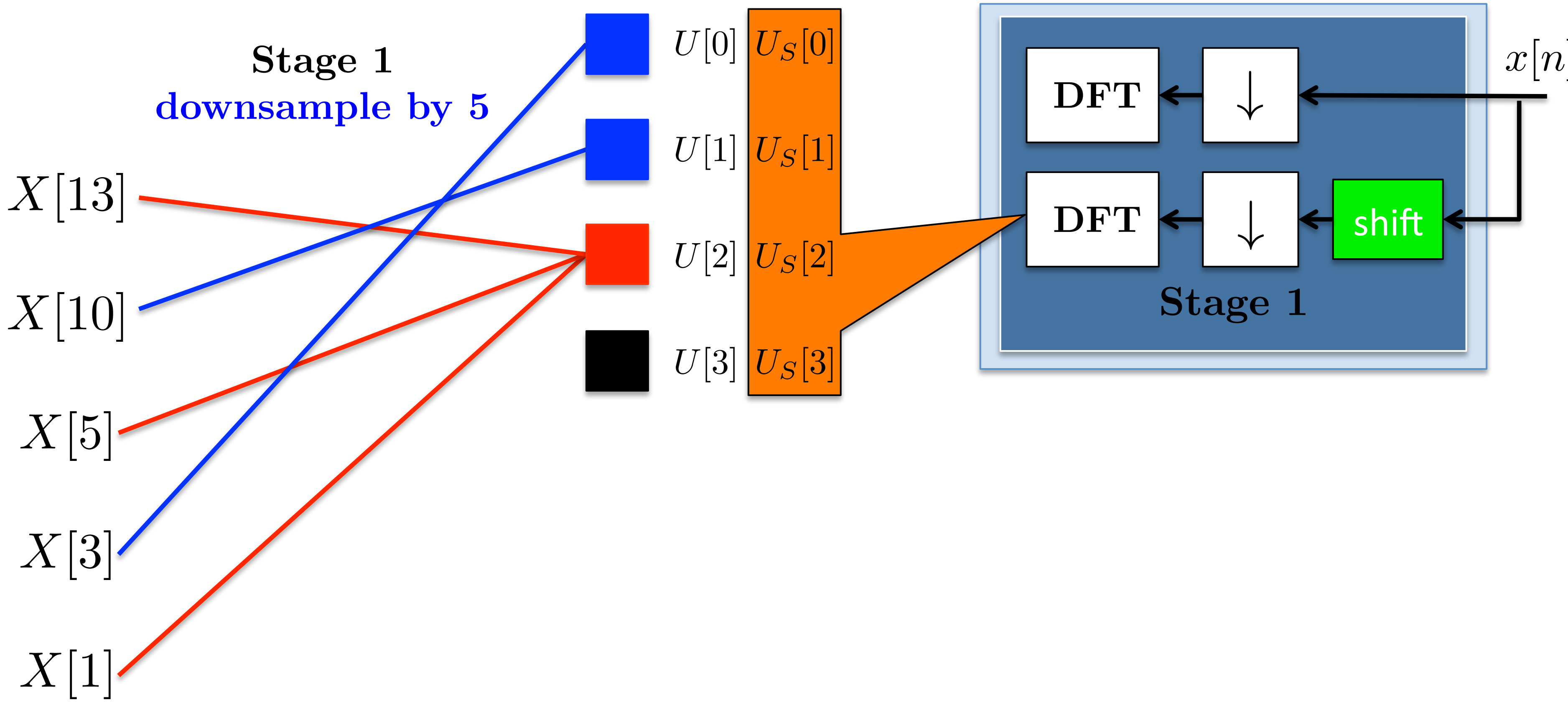


Kay, Steven. "A fast and accurate single frequency estimator." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 37.12 (1989): 1987-1990.

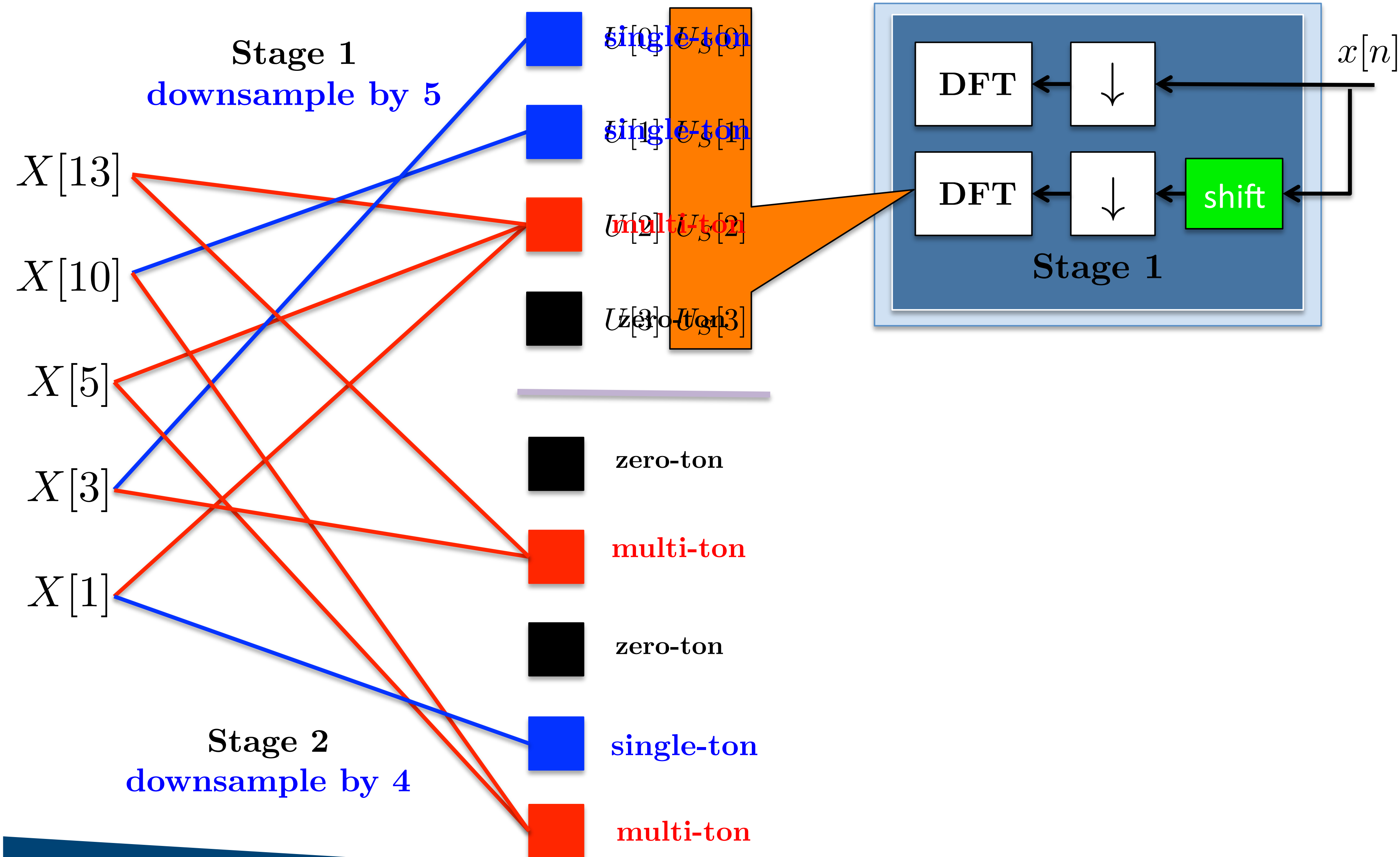
Computing Sparse DFT | Main Idea



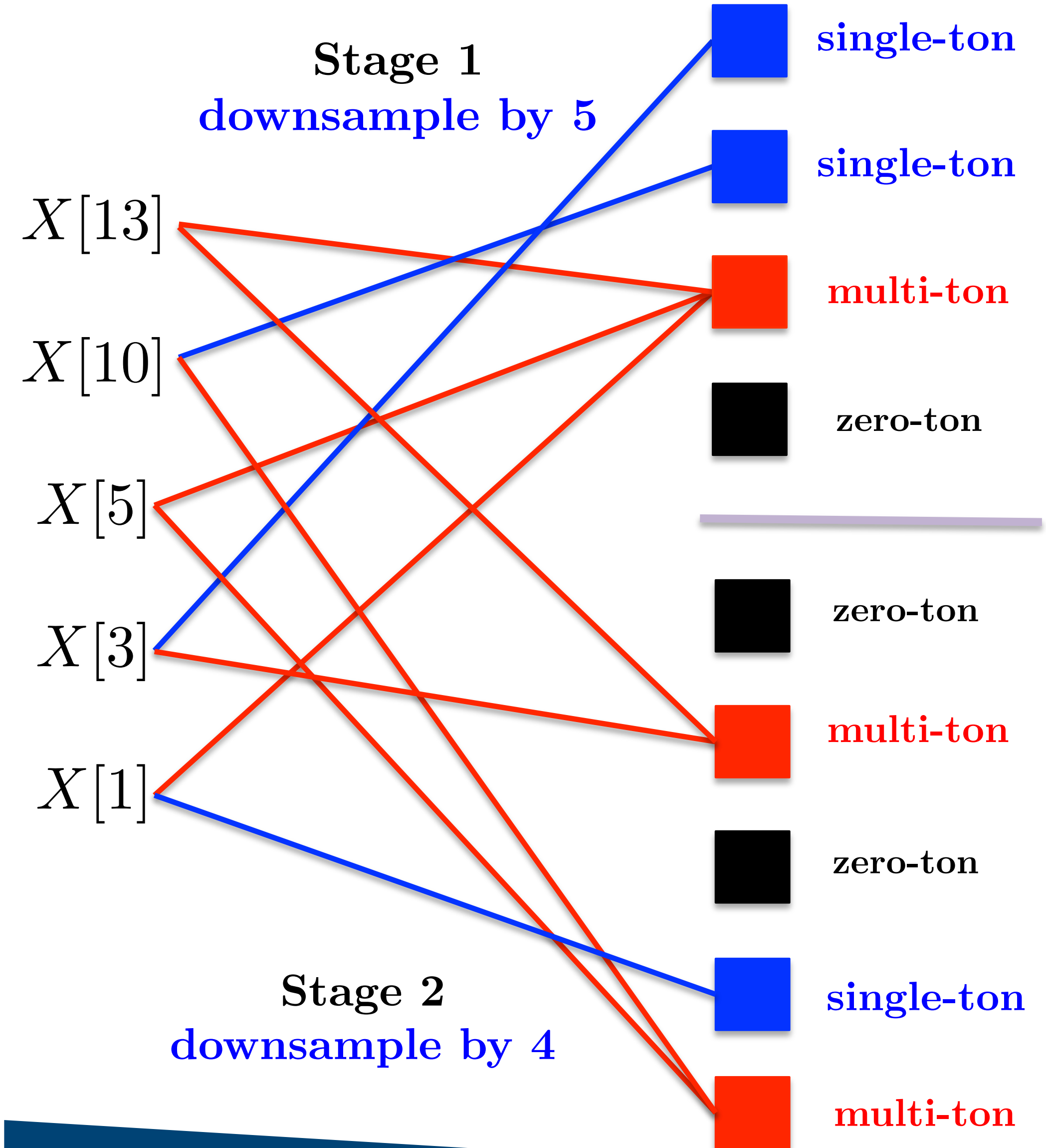
Computing Sparse DFT | Main Idea

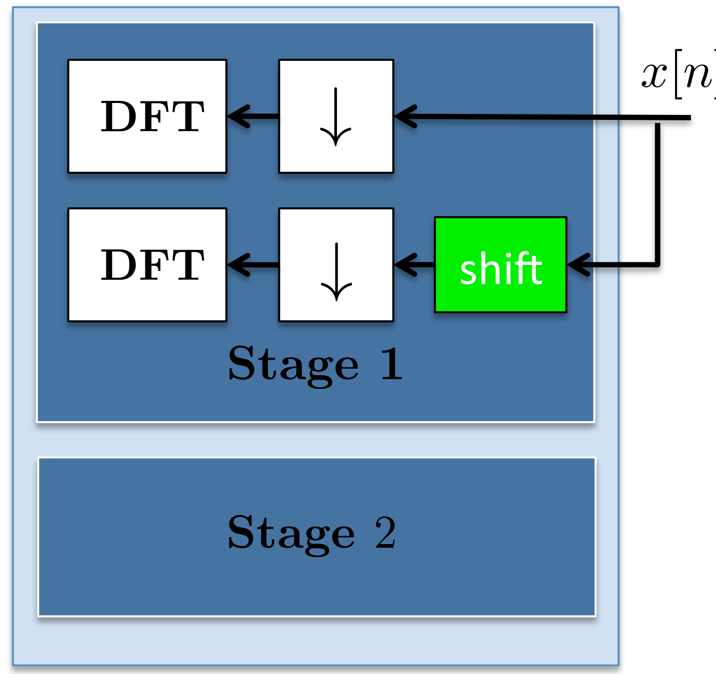


Computing Sparse DFT | Main Idea

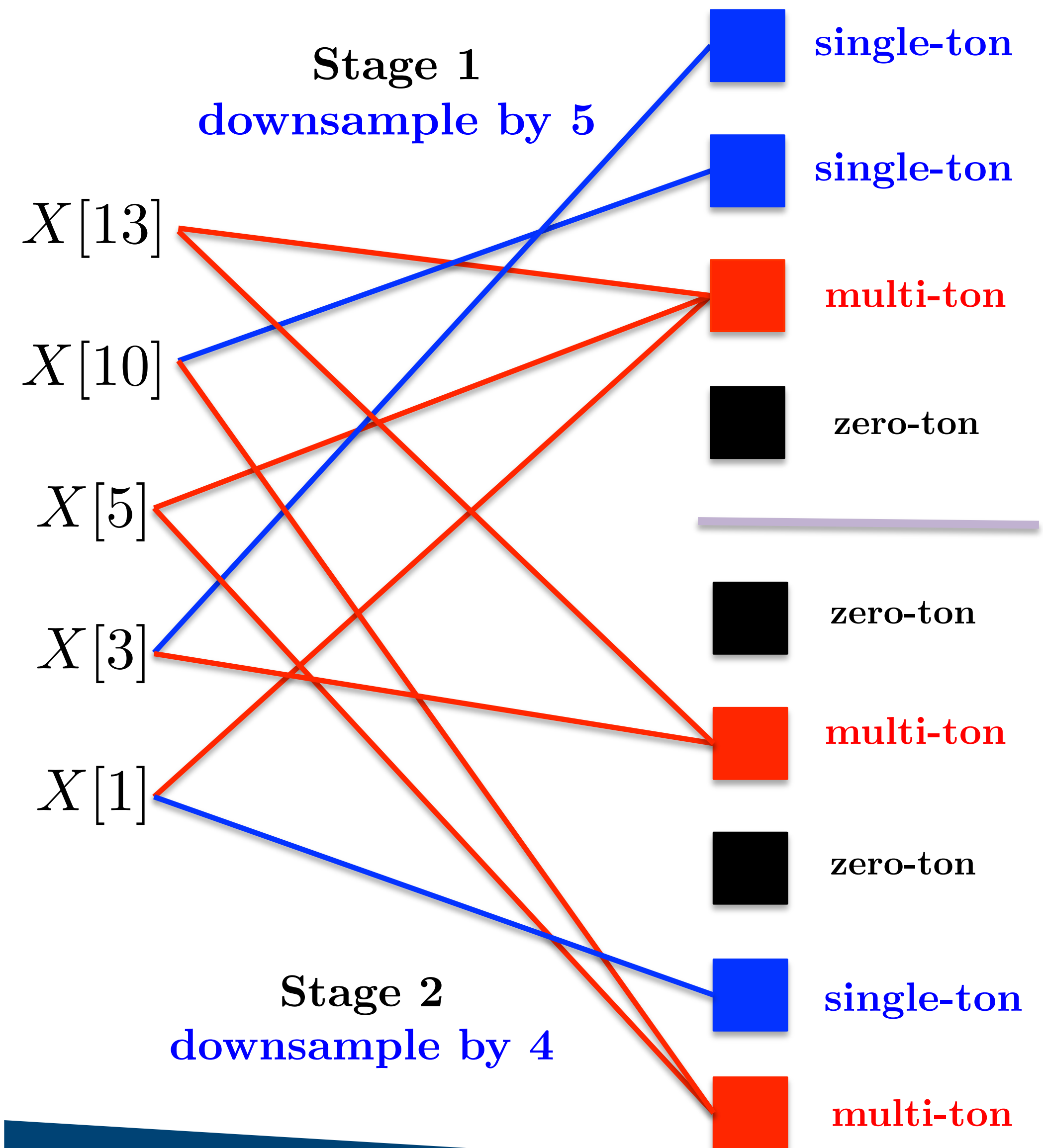


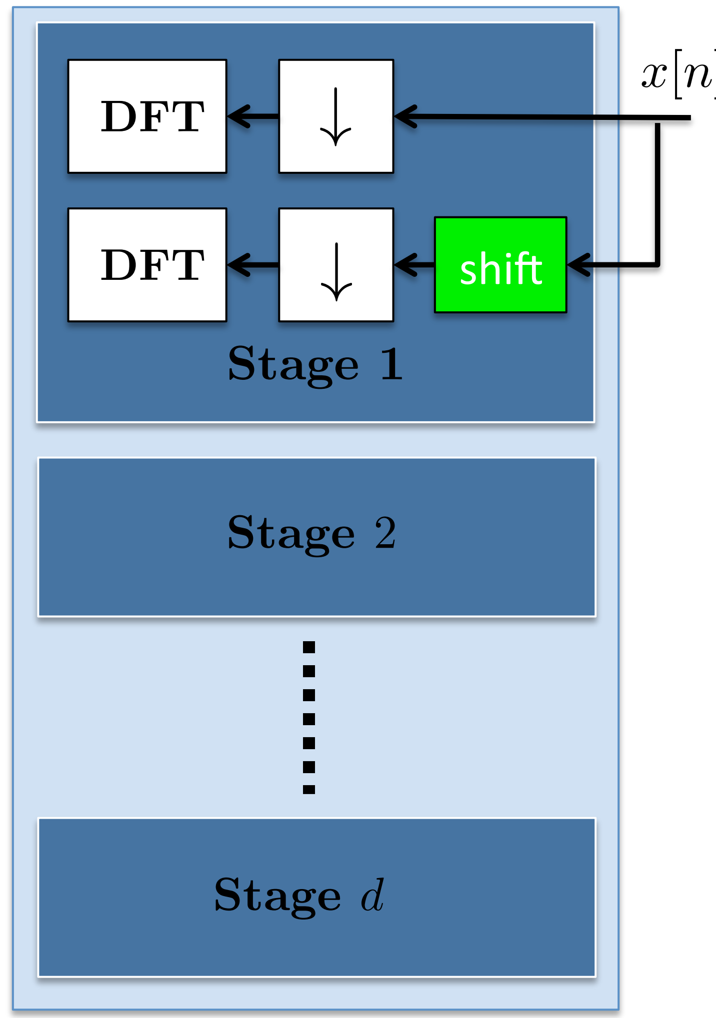
Computing Sparse DFT | Main Idea



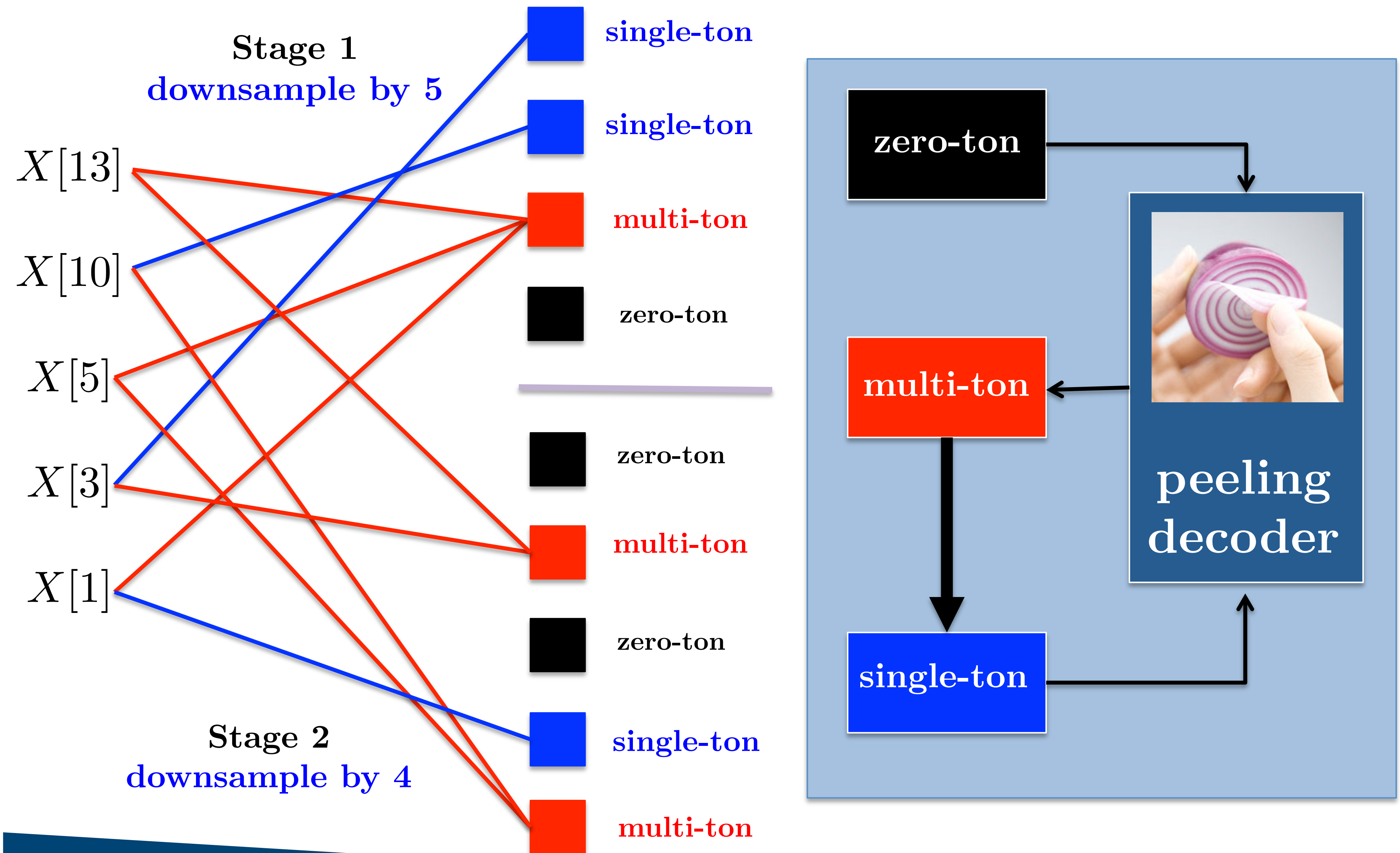


Computing Sparse DFT | Main Idea

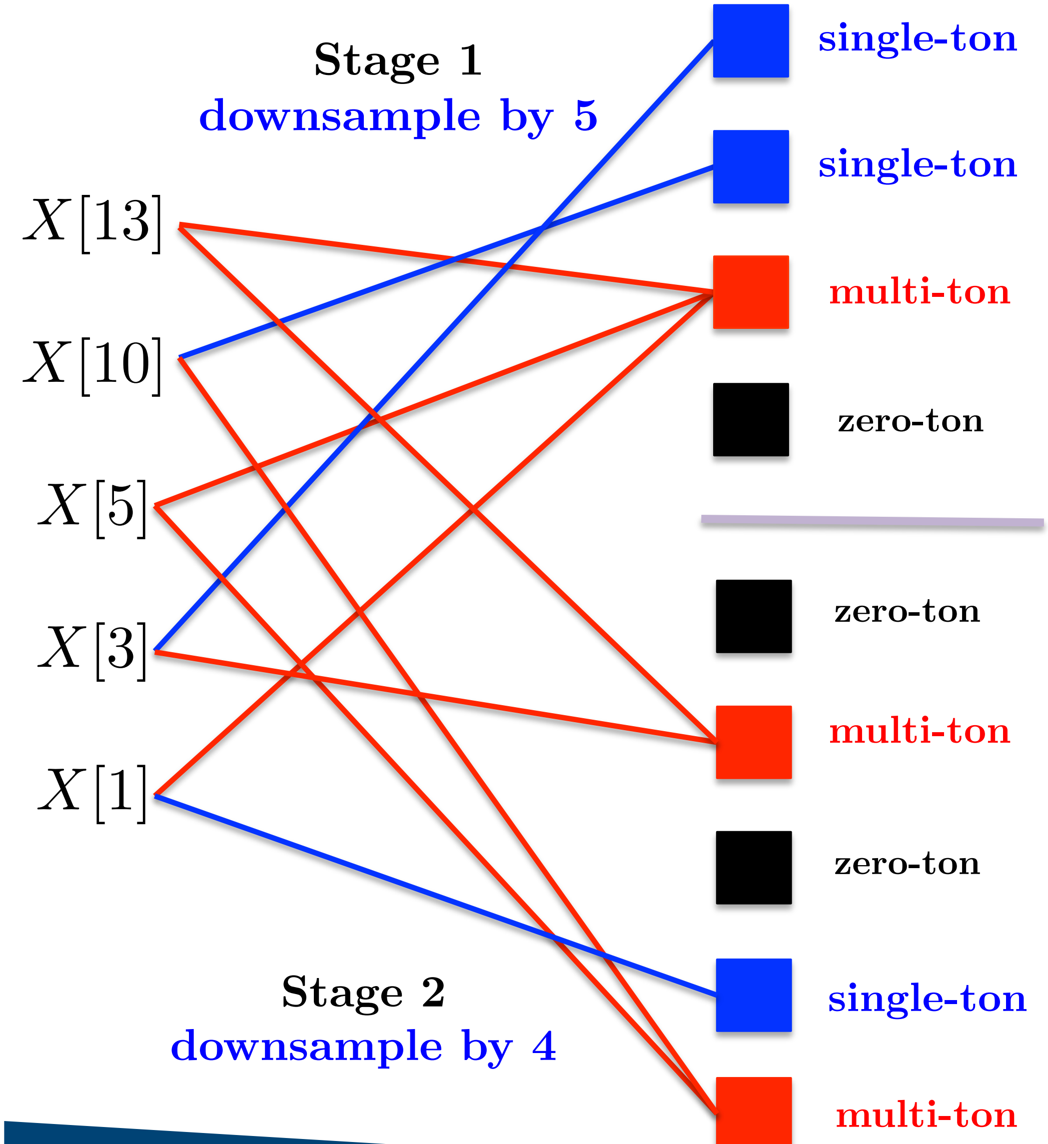




Computing Sparse DFT | Main Idea

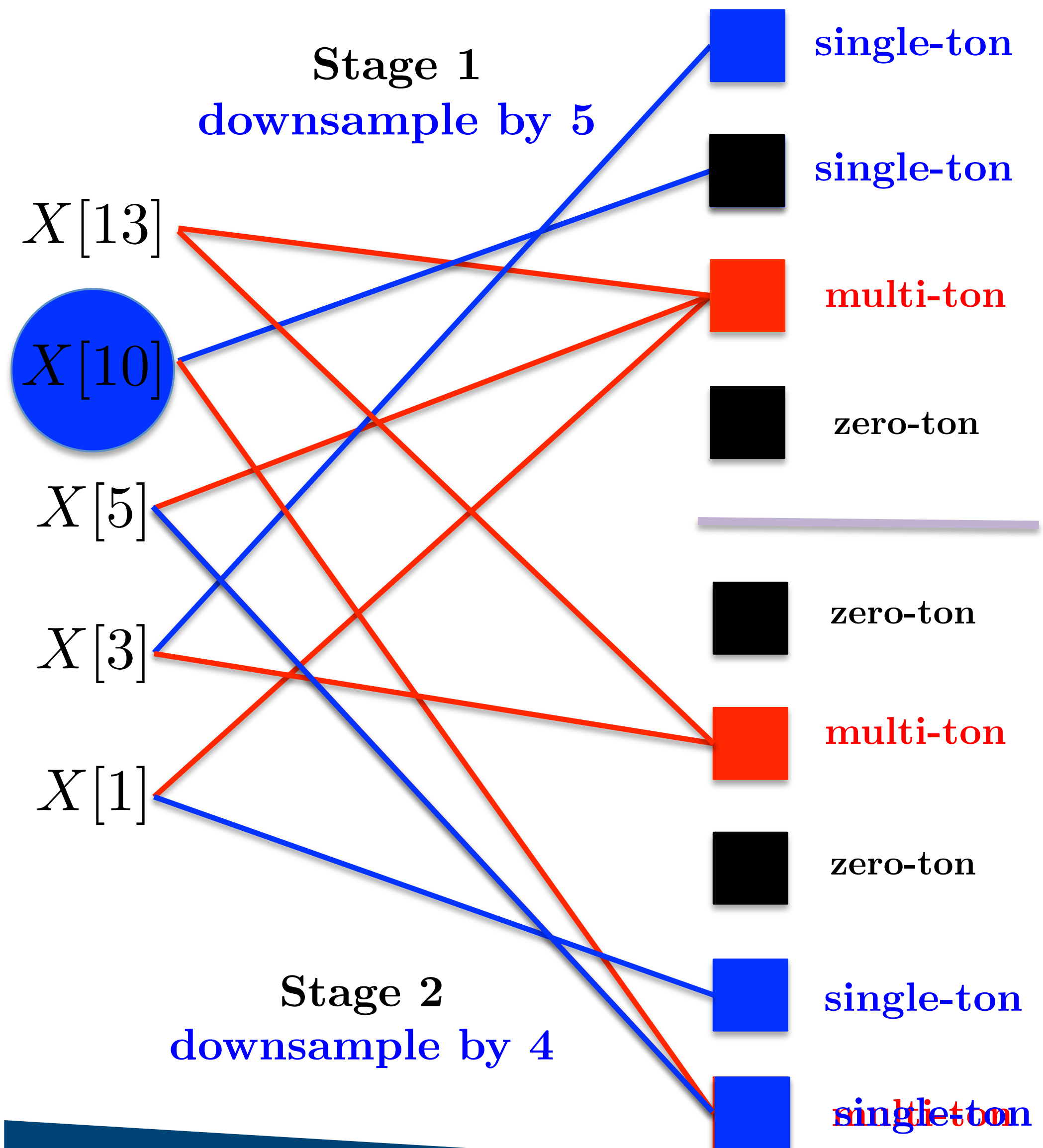


Computing Sparse DFT | Main Idea



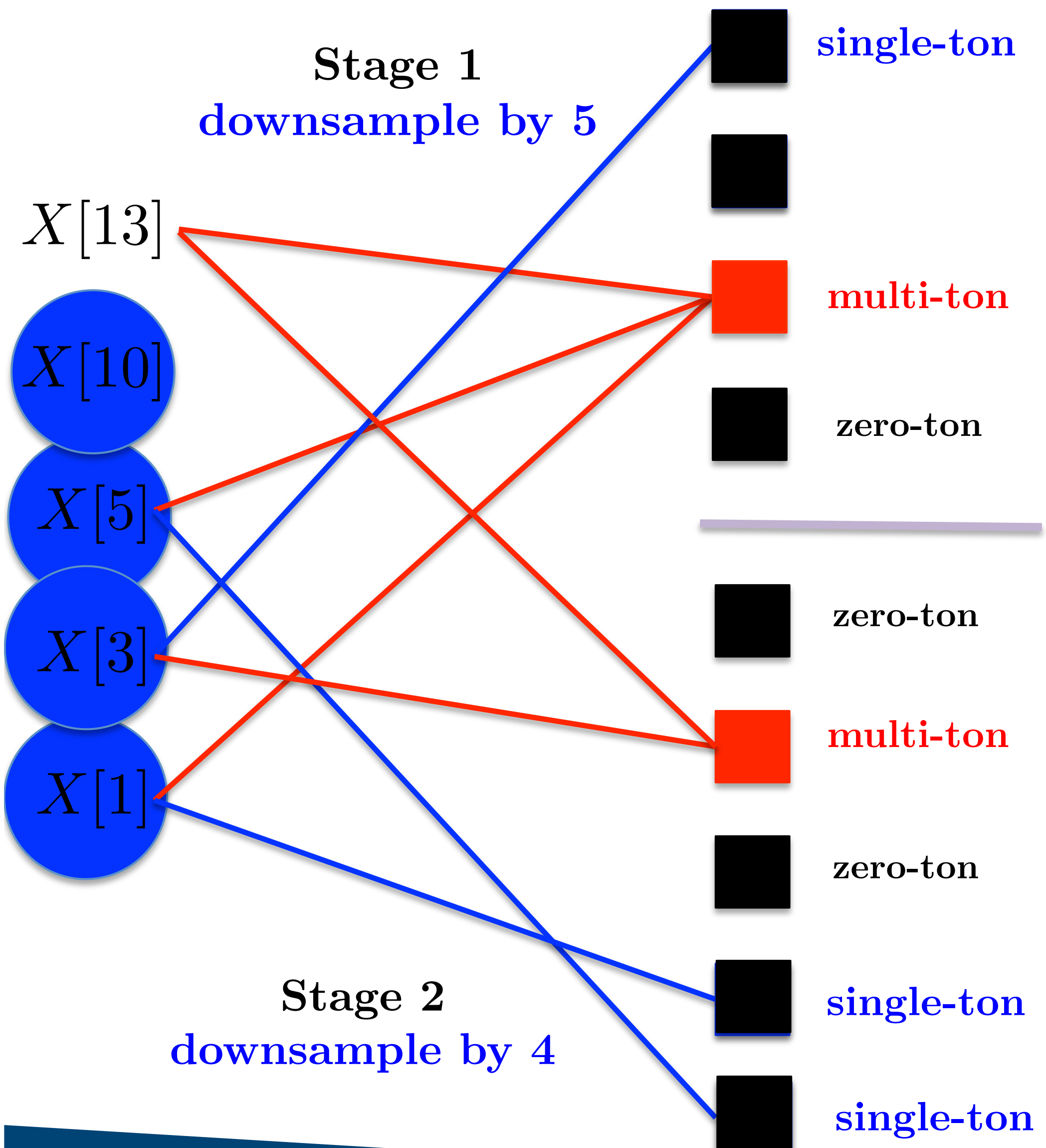
peeling decoder

Computing Sparse DFT | Main Idea



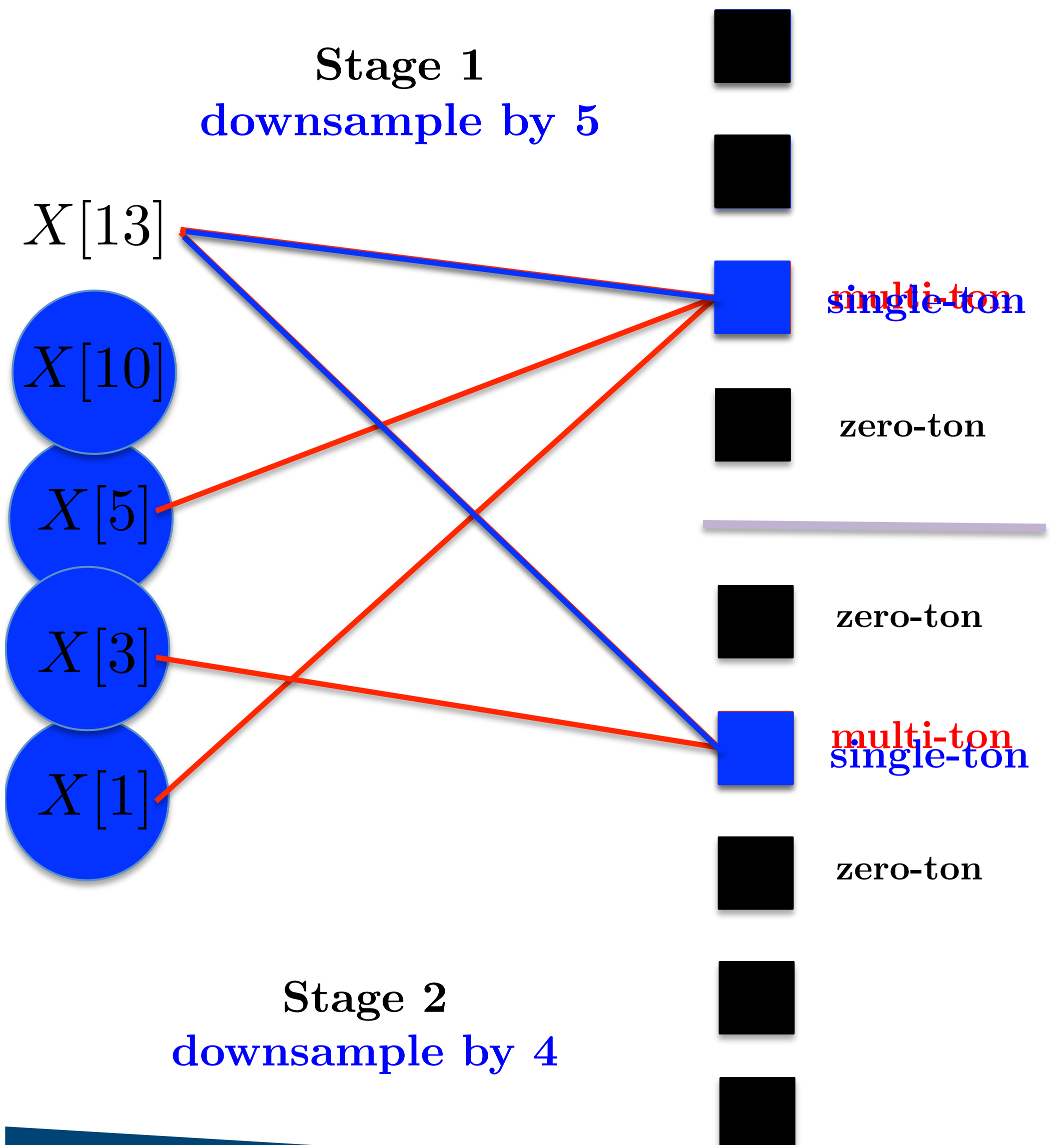
peeling decoder

Computing Sparse DFT | Main Idea



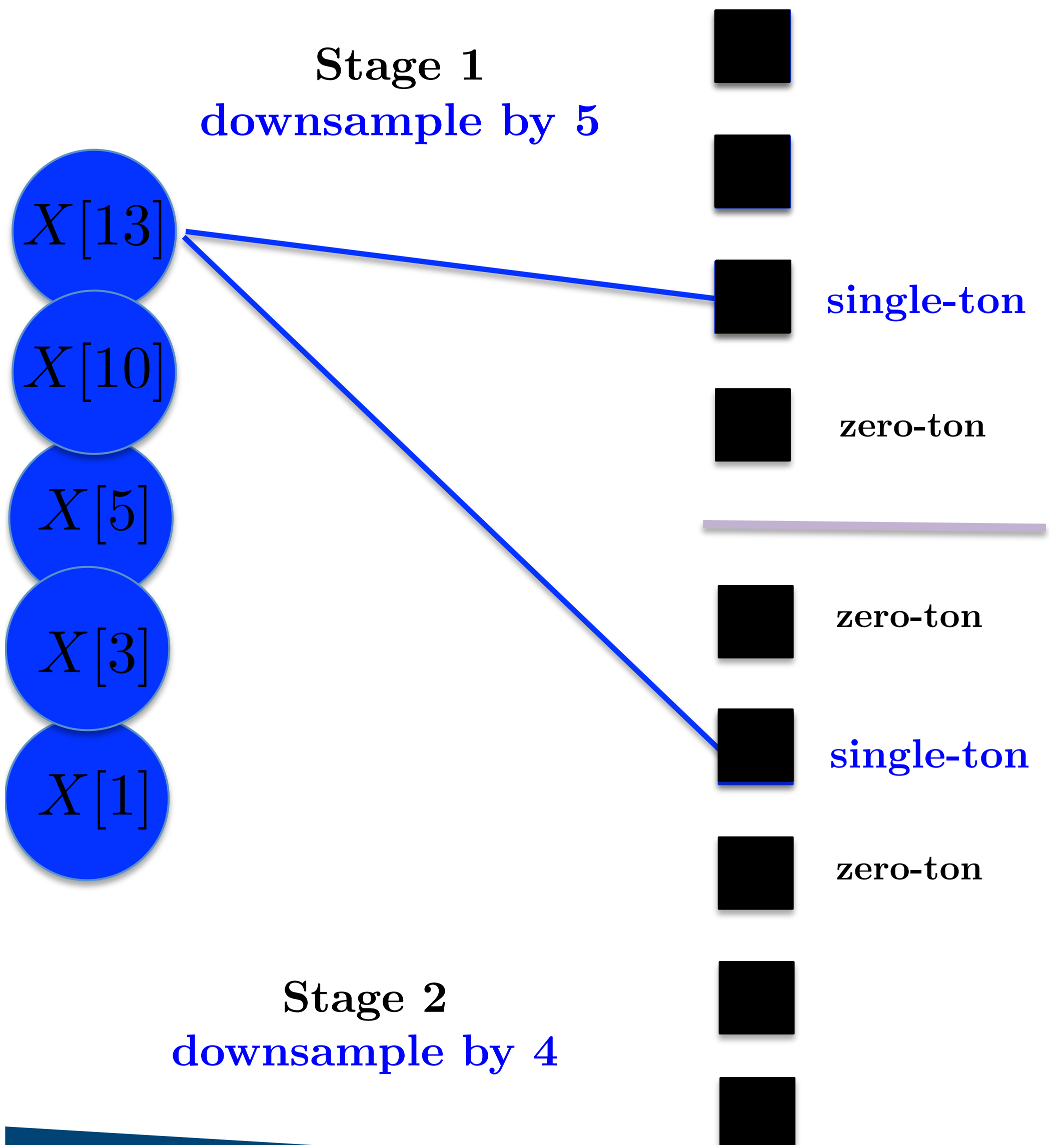
peeling decoder

Computing Sparse DFT | Main Idea



peeling decoder

Computing Sparse DFT | Main Idea



peeling decoder

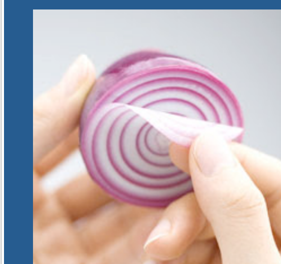
Computing Sparse DFT | Main Idea

Stage 1
downsample by 5

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

zero-ton
zero-ton
zero-ton
zero-ton

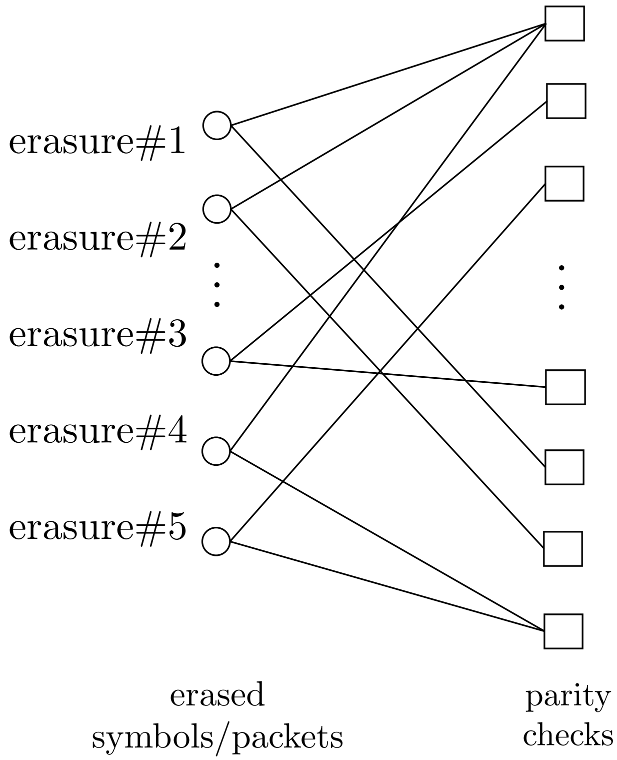
zero-ton
zero-ton
zero-ton
zero-ton

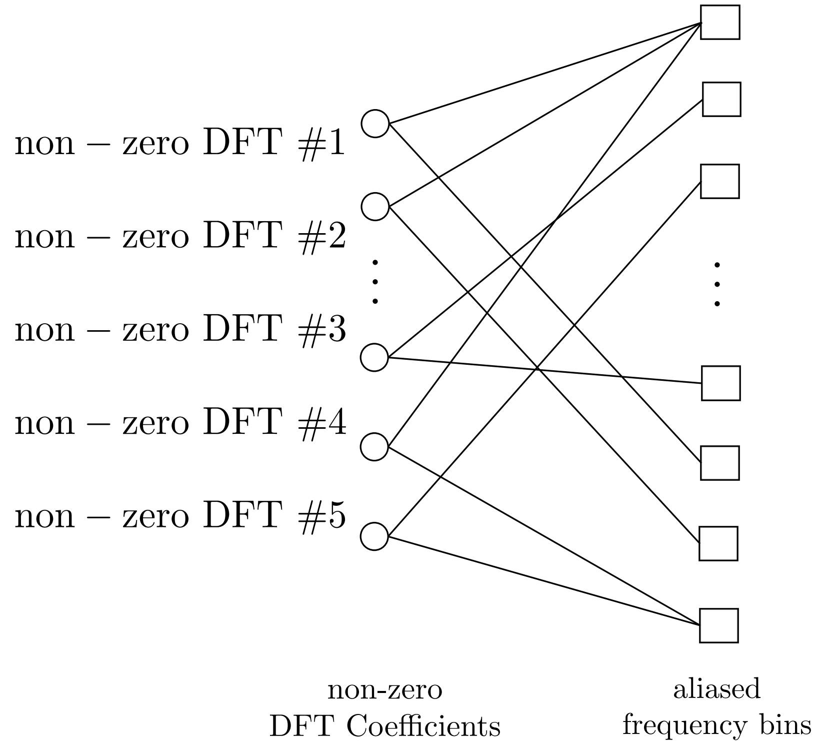


peeling decoder

How to induce good graphs that will work?

Sparse DFT Computation = Decoding over Sparse Graphs

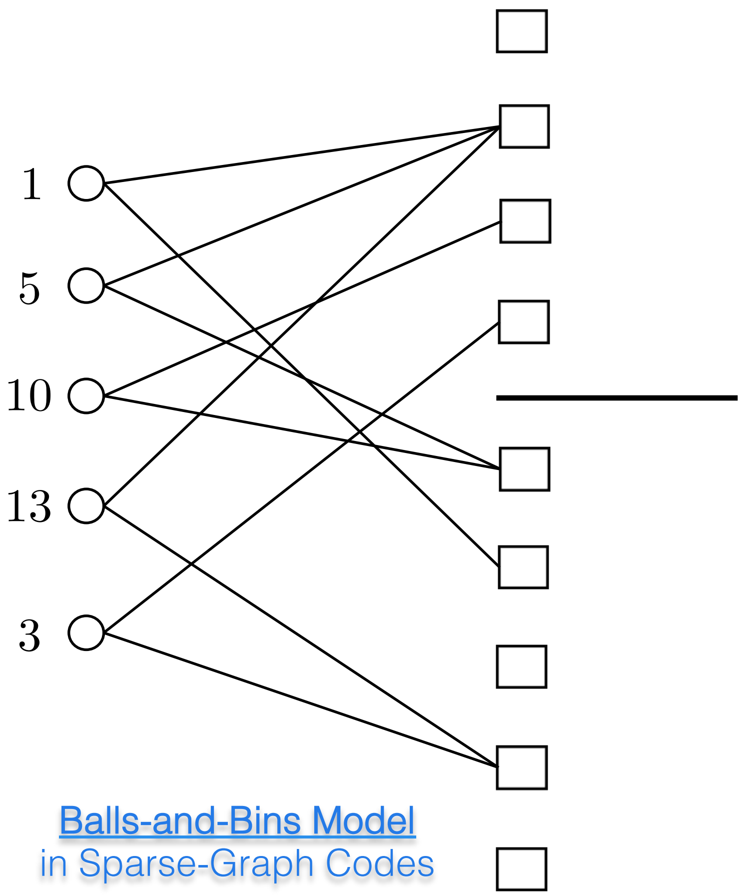




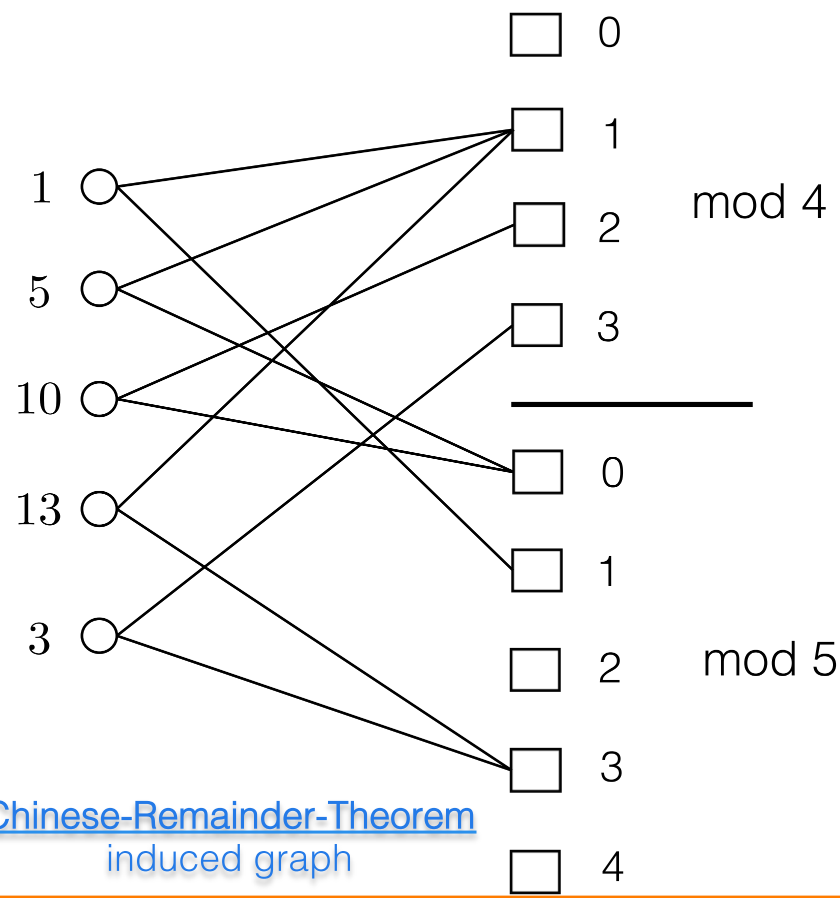
- **Explicit graph**: design well-understood.
- $(N - K)$ correctly received packets.
- K erased packets.
- Peeling decoder recovers values.

- **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.**

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

49

50

51

Chinese Remainder Theorem

$$49 \times 50 \times 51$$

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

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

49

50

51

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

49

50

51

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

49

50

51

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

49

50

51

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

$\boxed{49}$

$\boxed{50}$

$\boxed{51}$

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

49

50

51

Chinese Remainder Theorem

$49 \times 50 \times 51$

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

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

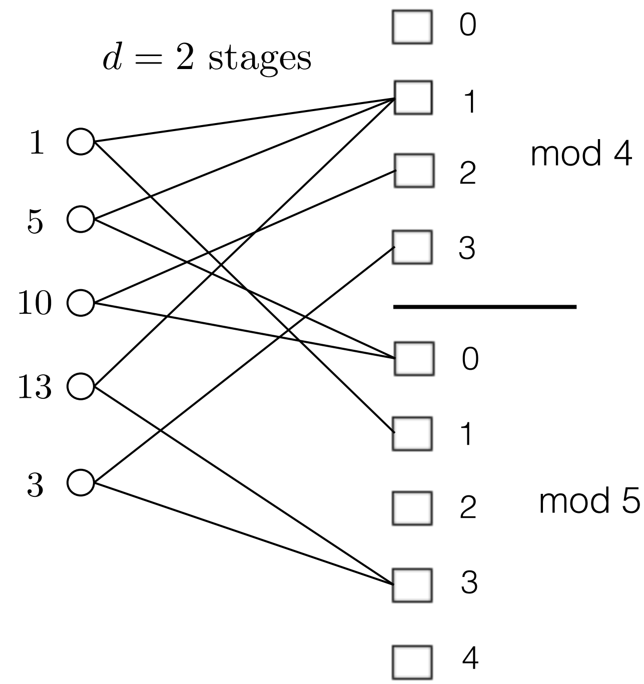
49

50

- $\delta = 2/3$ such that $K \approx 2500$

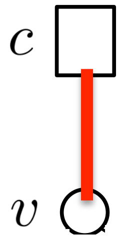
This can be generalized to any sparsity index between 0 and 1

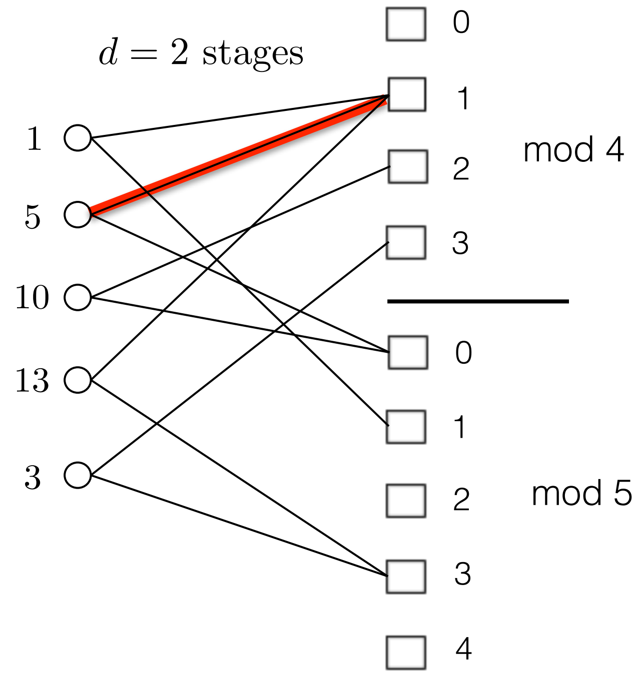
  - Stage 3: subsample by 51, keep 49×50



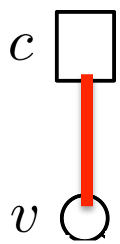
Goal: prove that the algorithm finishes Kd steps

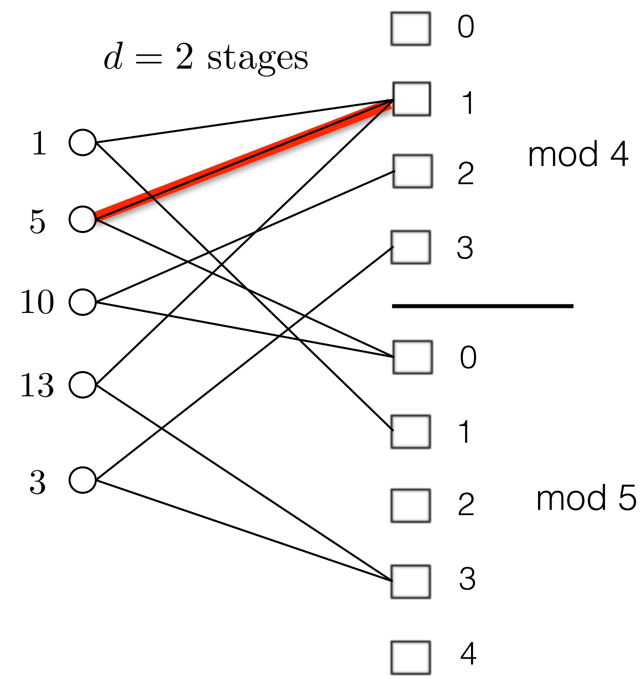
Kd edges to be removed



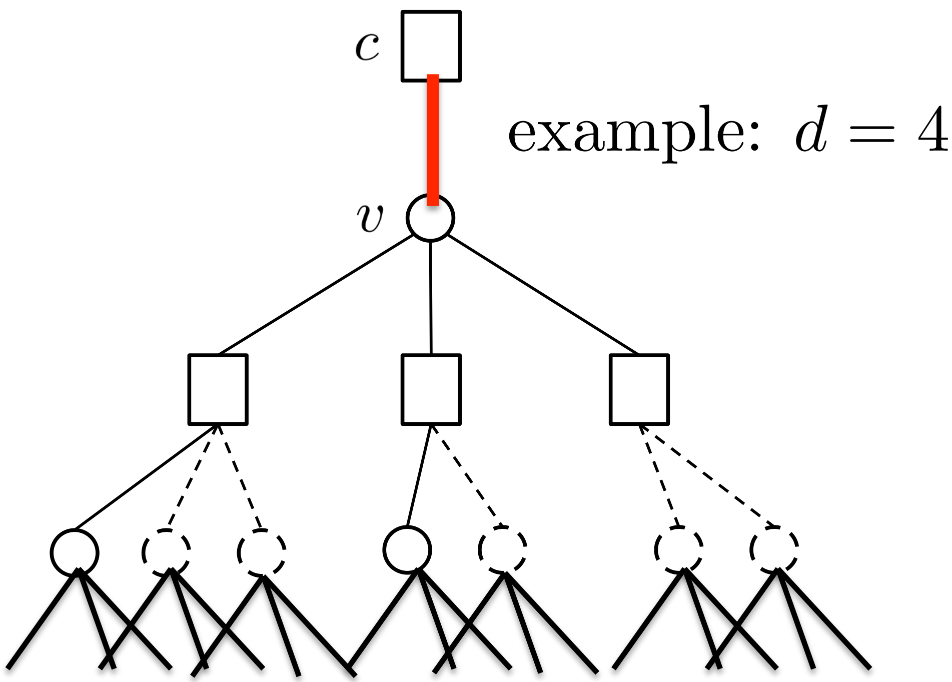


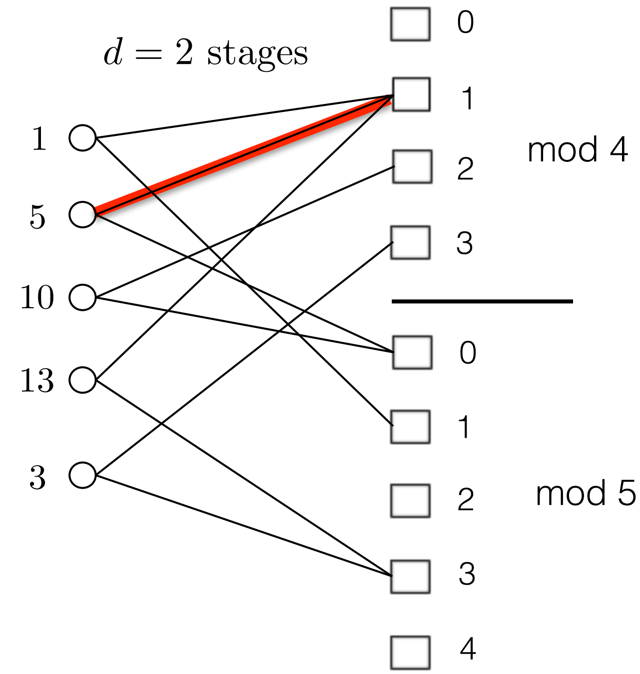
- Pick an arbitrary edge in the graph (c, v).



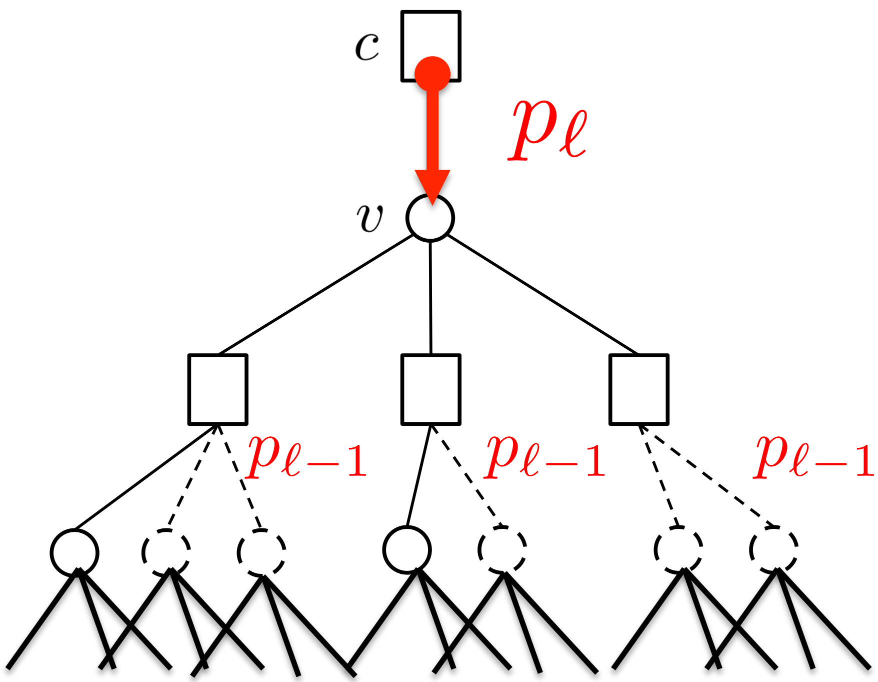


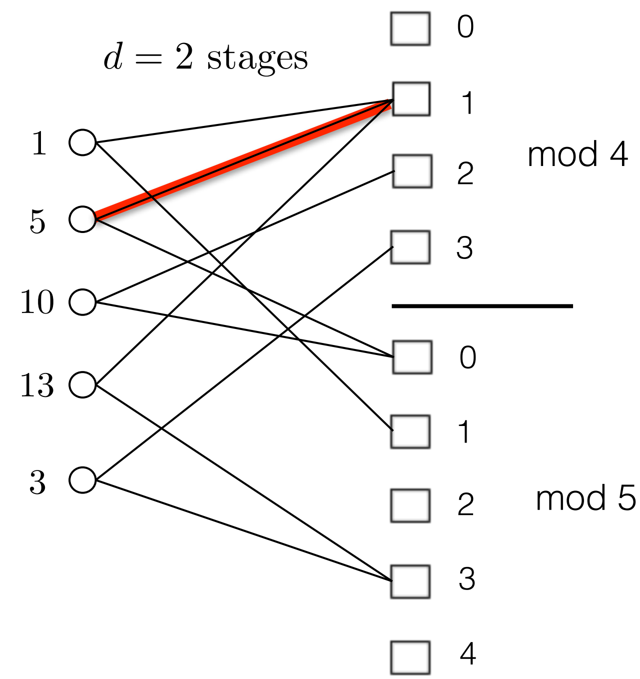
- Examine its directed neighborhood at depth-2ℓ





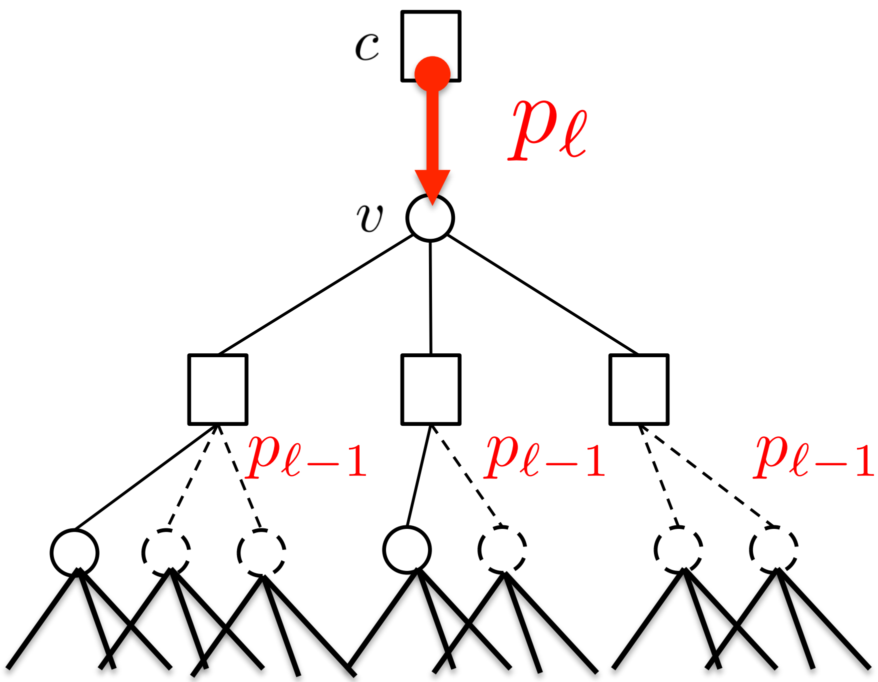
- Examine its directed neighborhood at depth-2ℓ

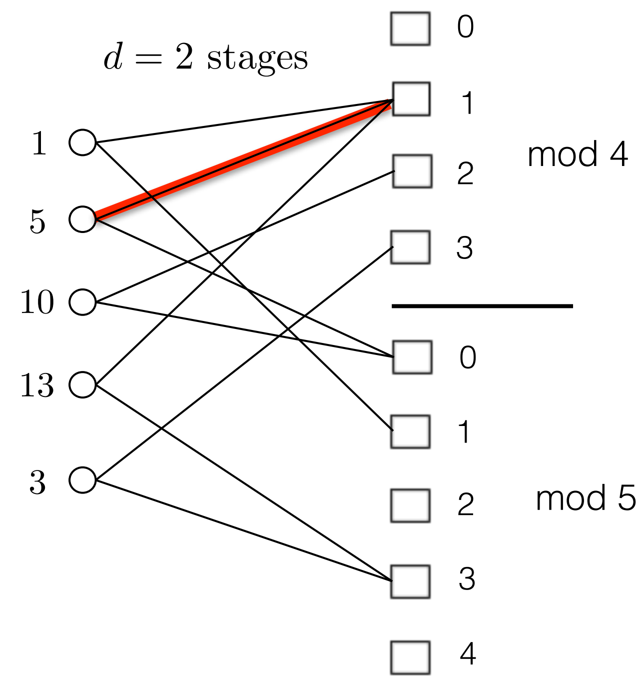




p_ℓ = probability of being **present** at depth-2ℓ

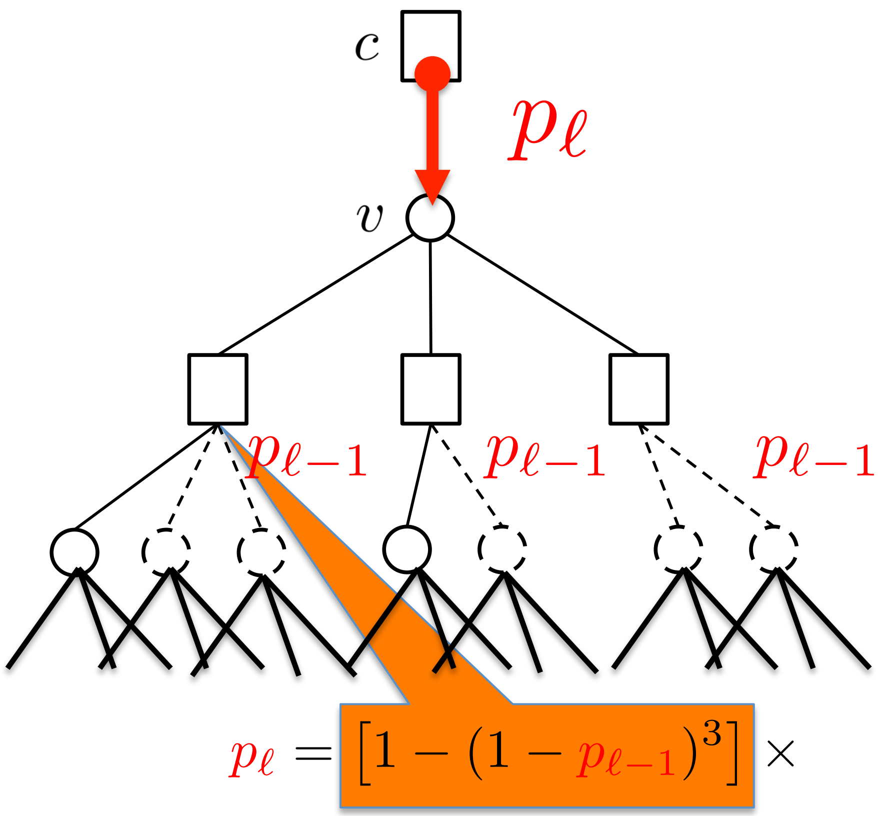
Density Evolution | A Hitchhiker's Guide

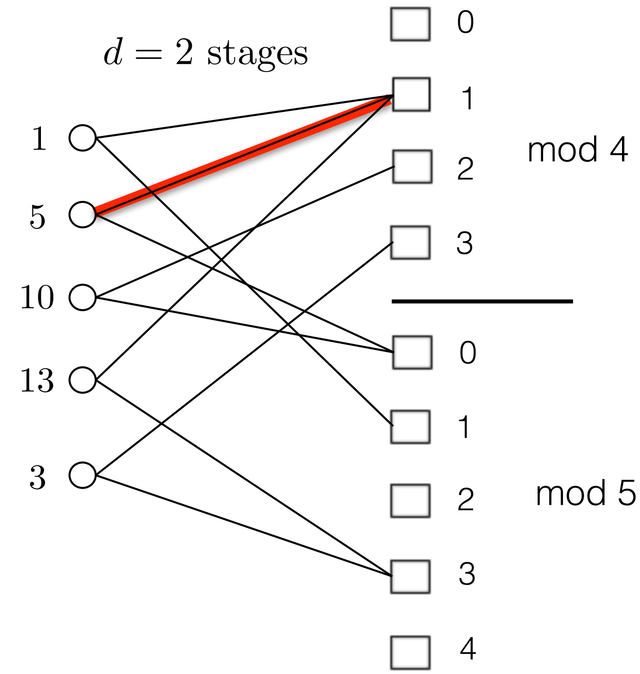




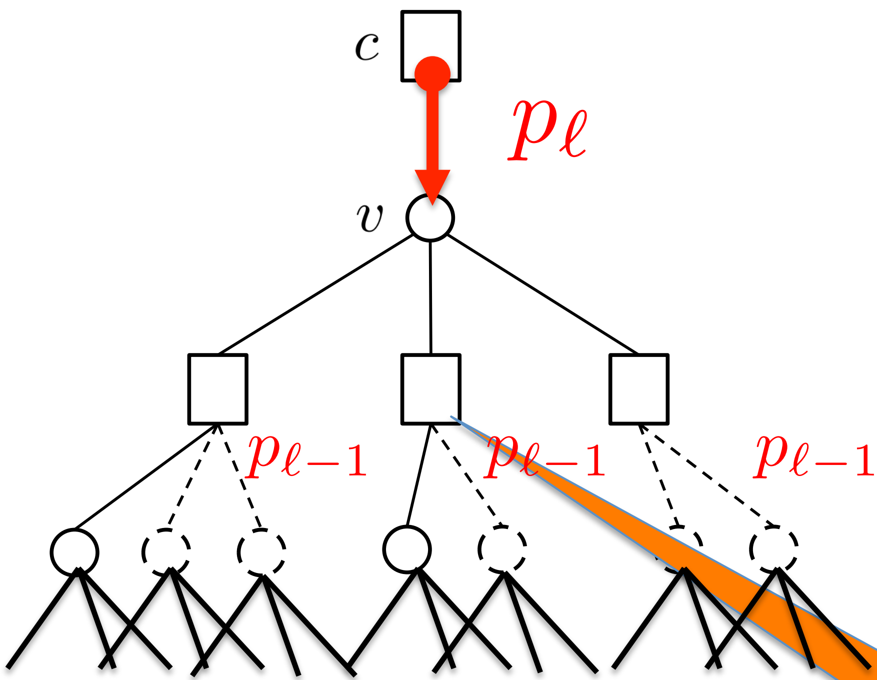
$p_\ell =$

Density Evolution | A Hitchhiker's Guide

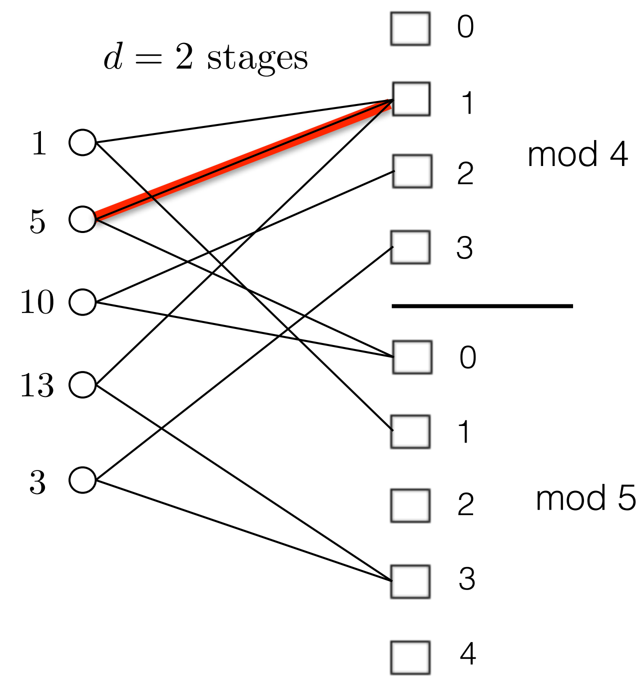




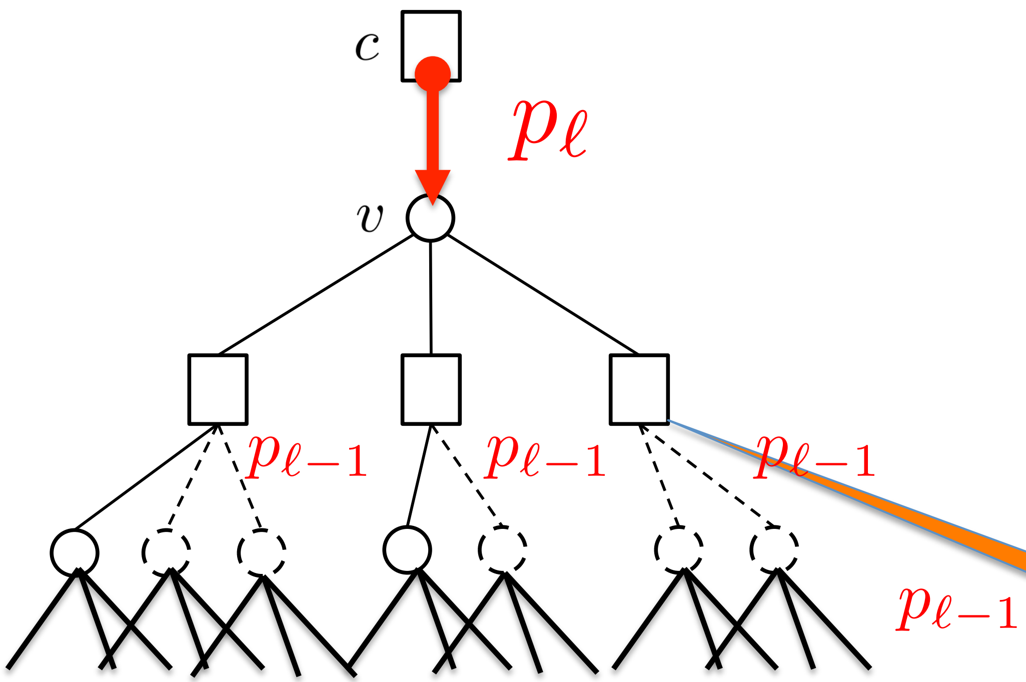
Density Evolution | A Hitchhiker's Guide

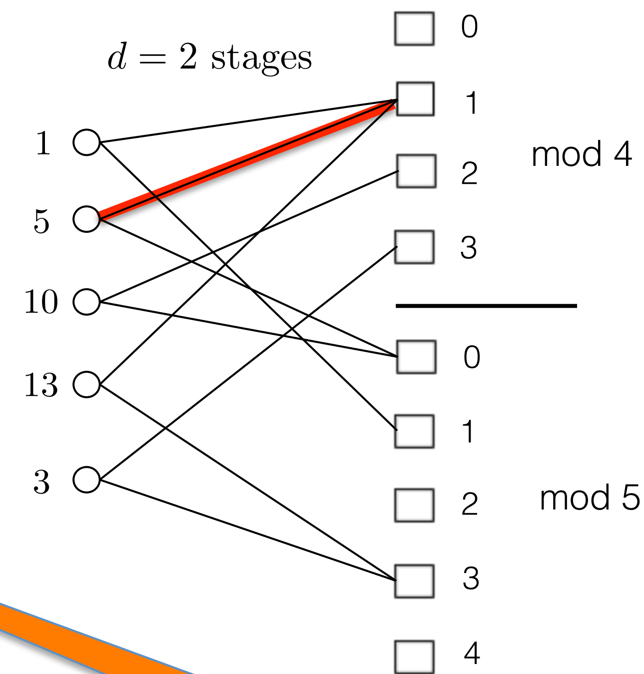


$$p_\ell = \left[1-(1-p_{\ell-1})^3\right] \times \boxed{\left[1-(1-p_{\ell-1})^2\right]} \times$$

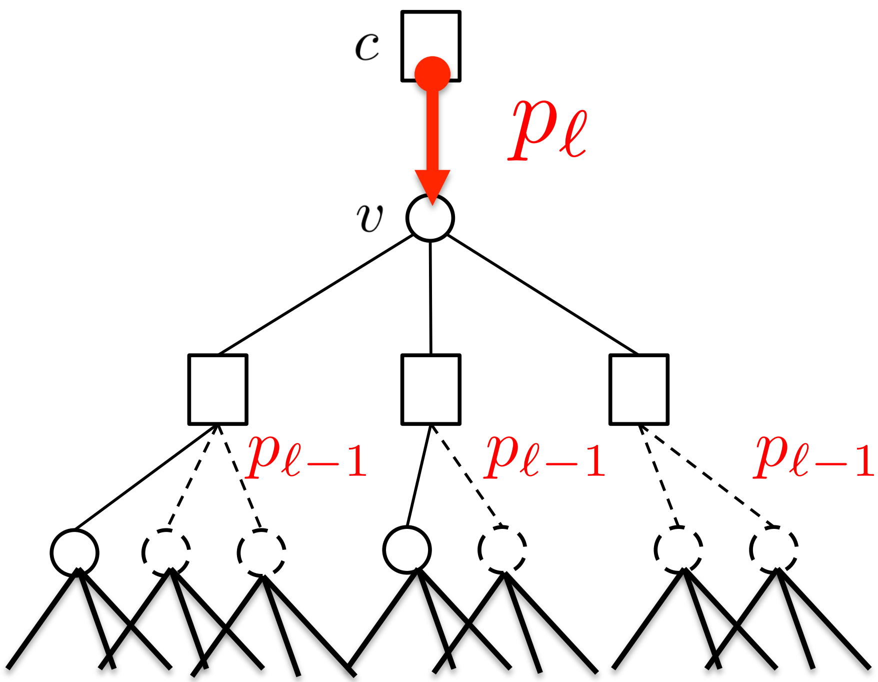


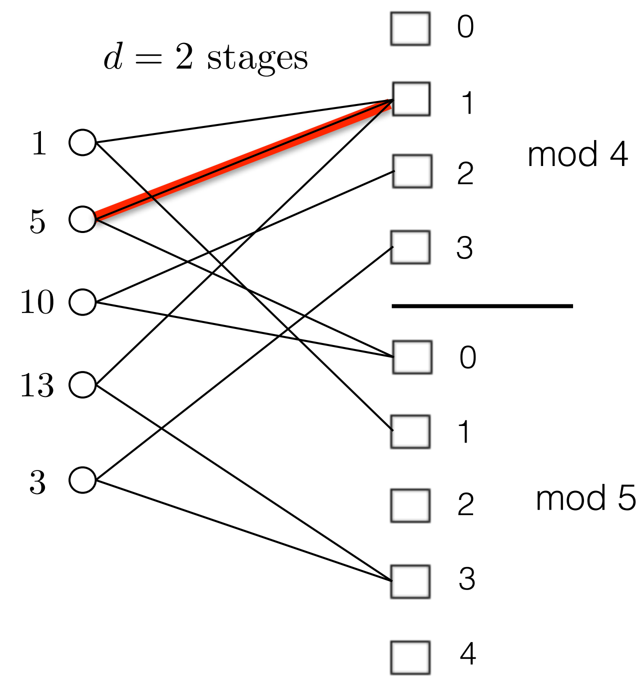
Density Evolution | A Hitchhiker's Guide





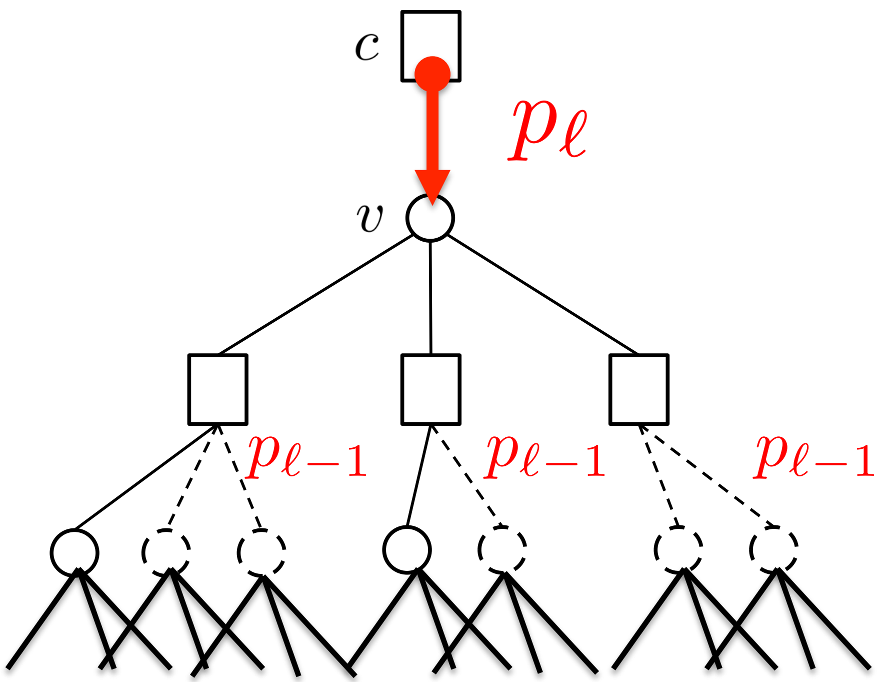
$$p_\ell = \left[1-(1-p_{\ell-1})^3\right] \times \left[1-(1-p_{\ell-1})^2\right] \times \boxed{\left[1-(1-p_{\ell-1})^2\right]}$$

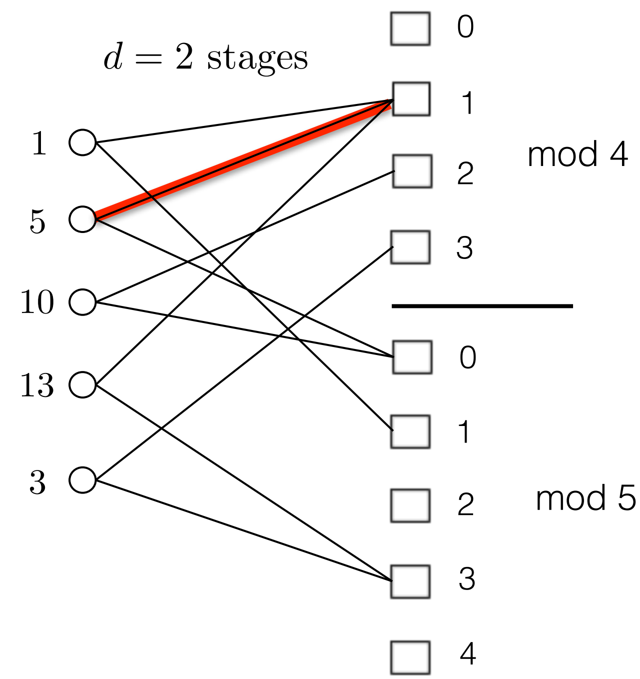




$$p_\ell = \left[1 - (1 - p_{\ell-1})^3\right] \times \left[1 - (1 - p_{\ell-1})^2\right] \times \left[1 - (1 - p_{\ell-1})^2\right]$$

- It generalizes to d stages:





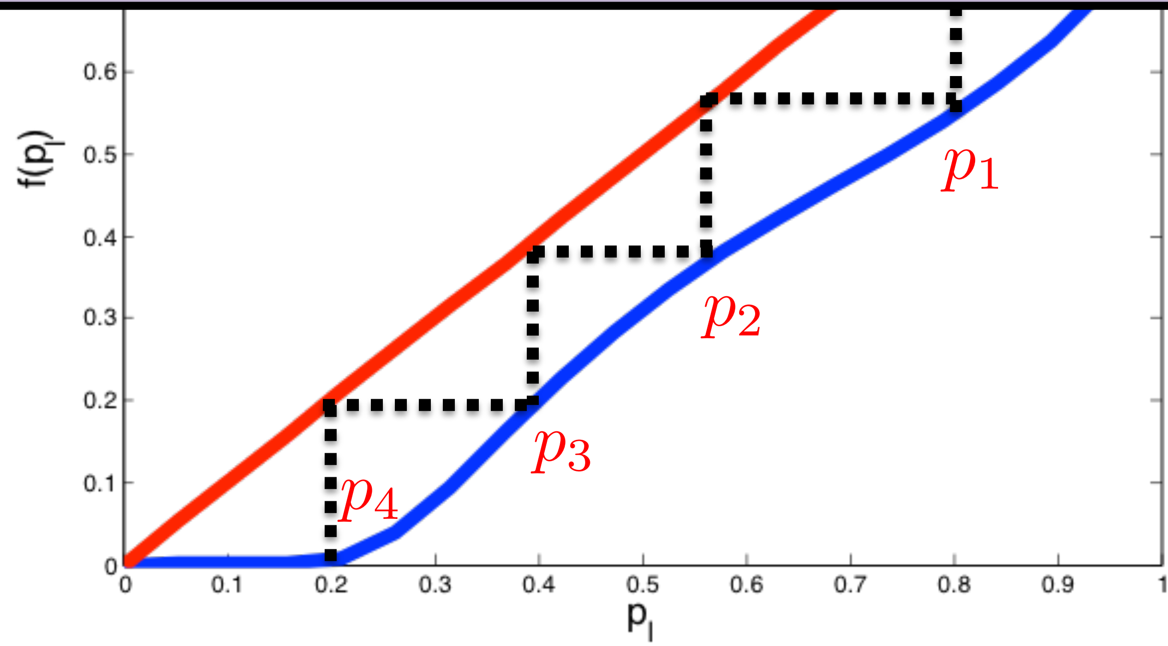
$$p_\ell = \left[1 - (1 - p_{\ell-1})^3\right] \times \left[1 - (1 - p_{\ell-1})^2\right] \times \left[1 - (1 - p_{\ell-1})^2\right]$$

- 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

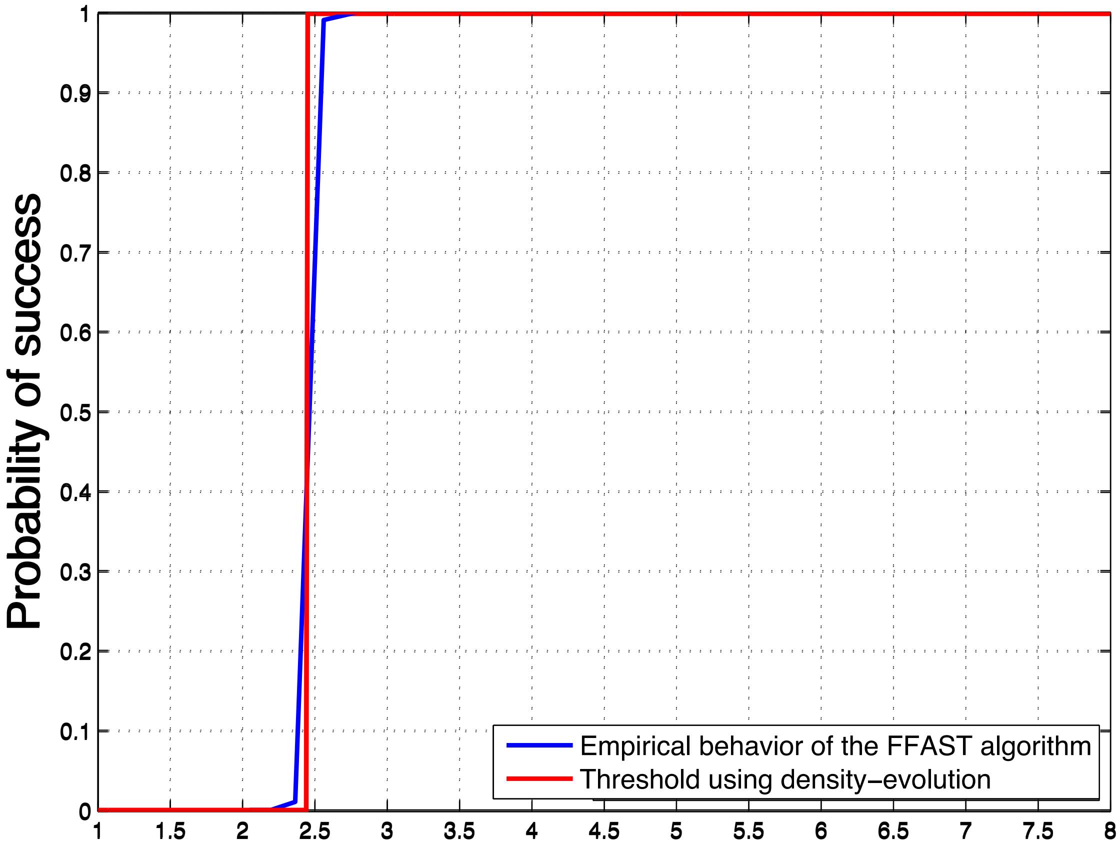
d	2	$\boxed{3}$	4	5	6
$M/2K$	2.0000	$\boxed{1.2219}$	1.2948	1.4250	1.5696



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

- K = sparsity
- M = # of samples
- d = # of stages

Sampling Rate | Noiseless Setting: Theory versus Practice



- $N = 7.7$ million
- $K = 400$
- $d = 3$ stages
- $M = 1248$ samples

Density Evolution

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

Theoretical threshold = 2 X 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_ℓ 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_ℓ can be made arbitrarily small

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

- **Performance Concentration**
  - overall average analysis $p_\ell^\star$ without tree assumption

$$|p_\ell^\star - p_\ell| < \epsilon_1, \ \forall \epsilon_1 > 0$$

  - actual performance concentrates around overall average analysis

$$\mathbb{P}\left(|\#\text{ of actual remaining edges} - Kdp_\ell^\star| > \epsilon_2\right) \to 0, \quad \forall \epsilon_2 > 0$$

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

?

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

Kdp_ℓ **edges** remain

Kd edges to be removed

- **Expander Graph**
  - the remaining Kdp_ℓ edges form an **expander graph**
  - expander graphs guarantee steady supplies of **single-tons**

success with high probability!

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

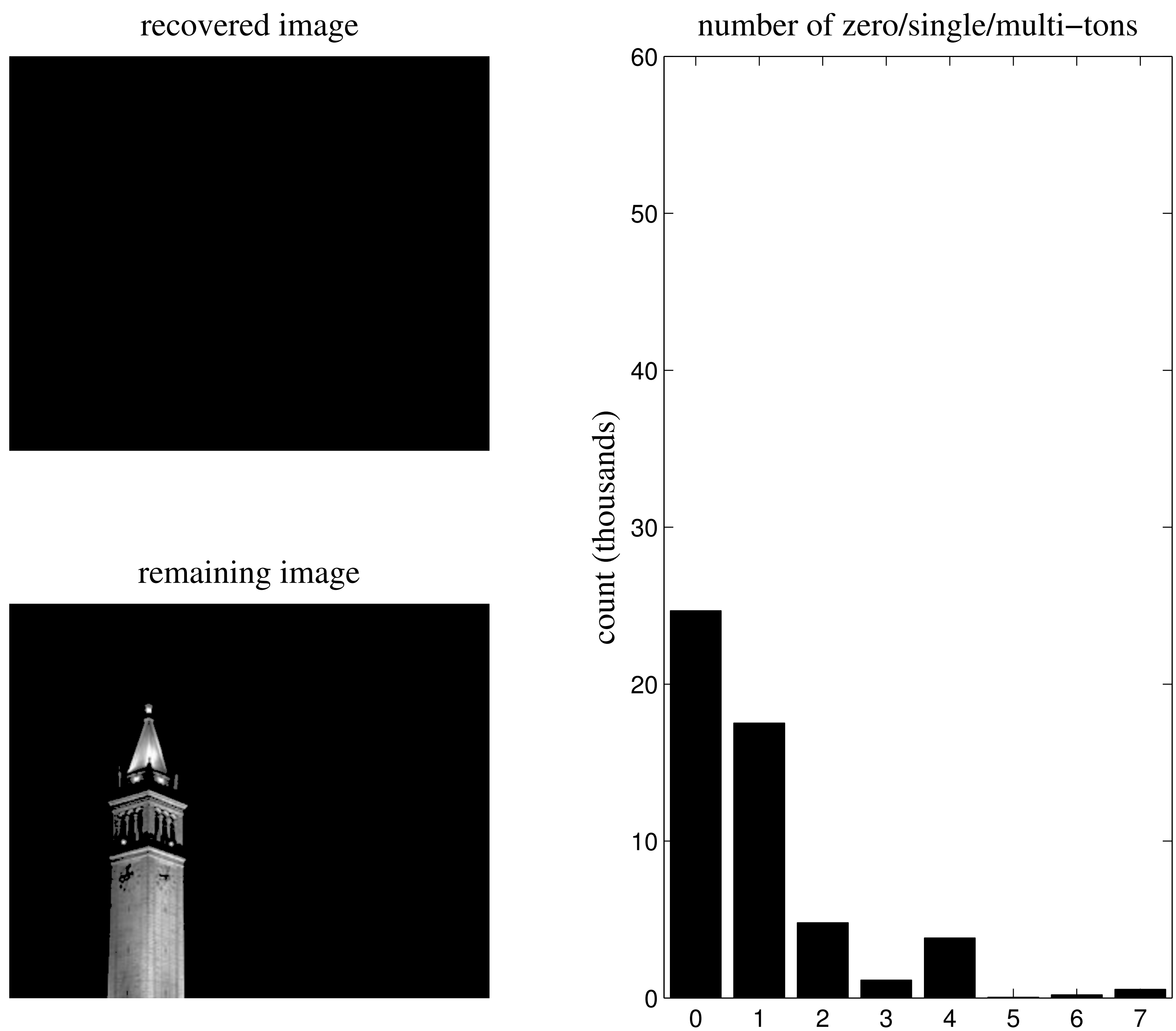
Kdp_ℓ **edges** remain

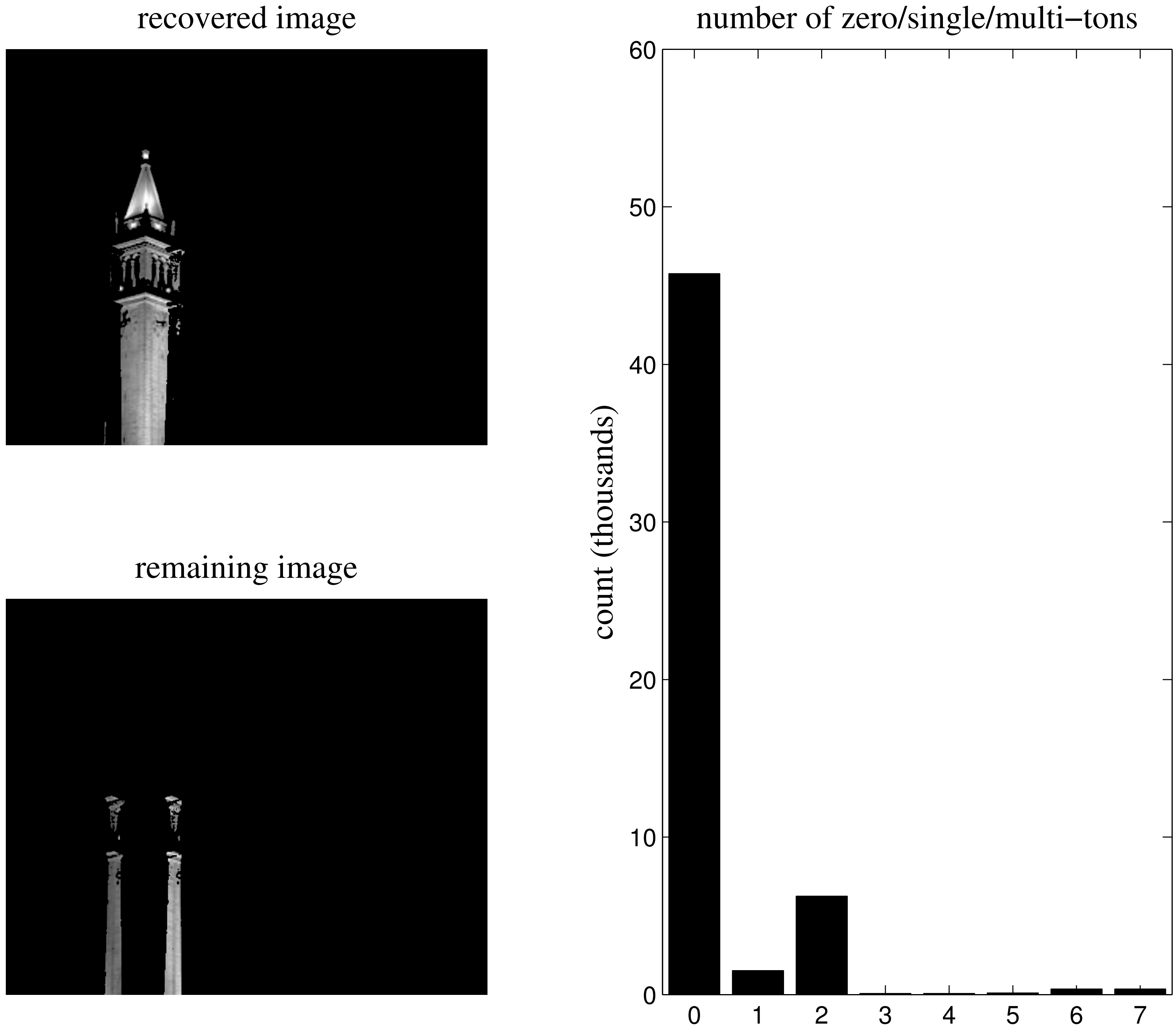
Kd edges to be removed

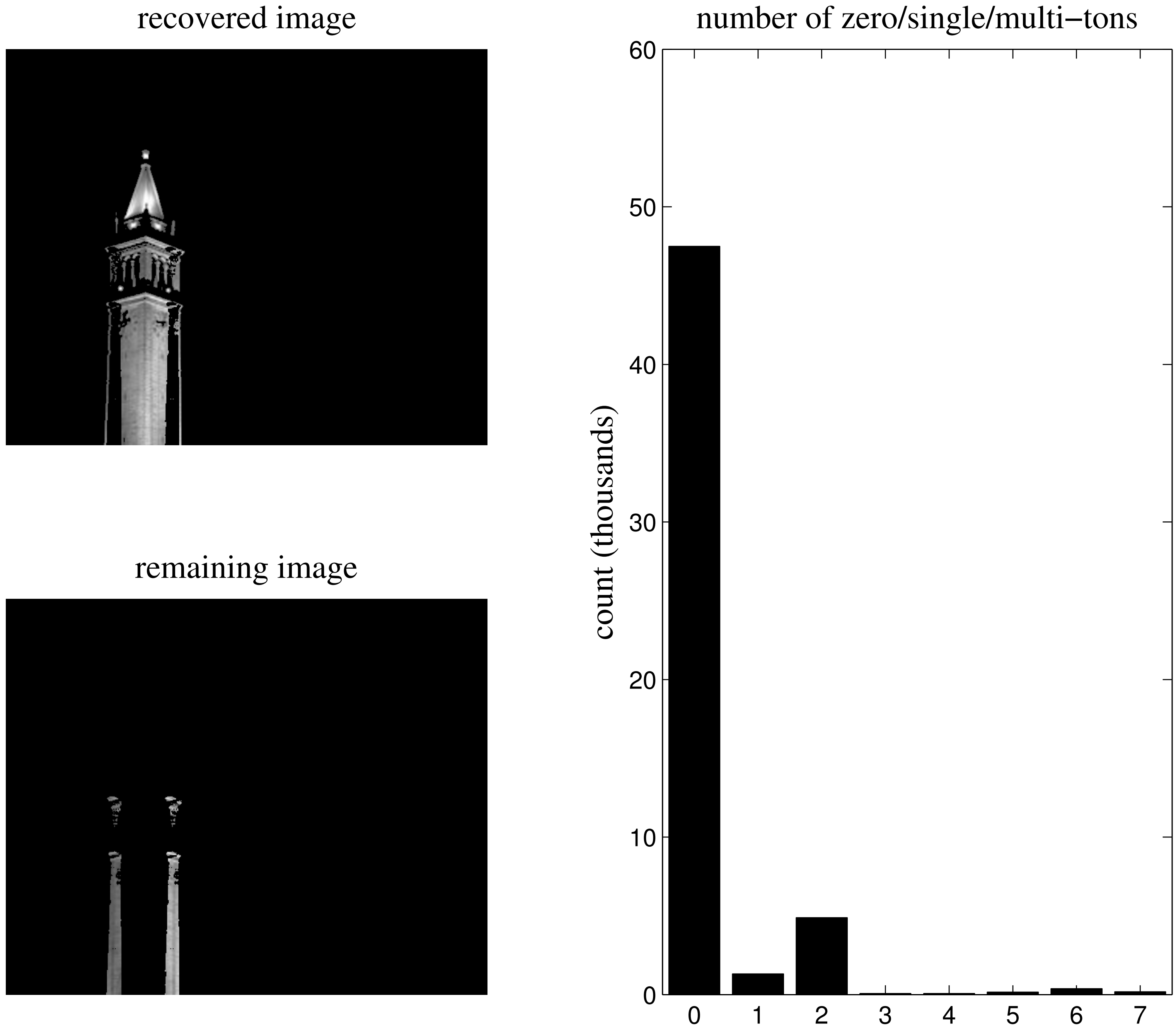
Peeling Performance | Numerical Examples

600×493

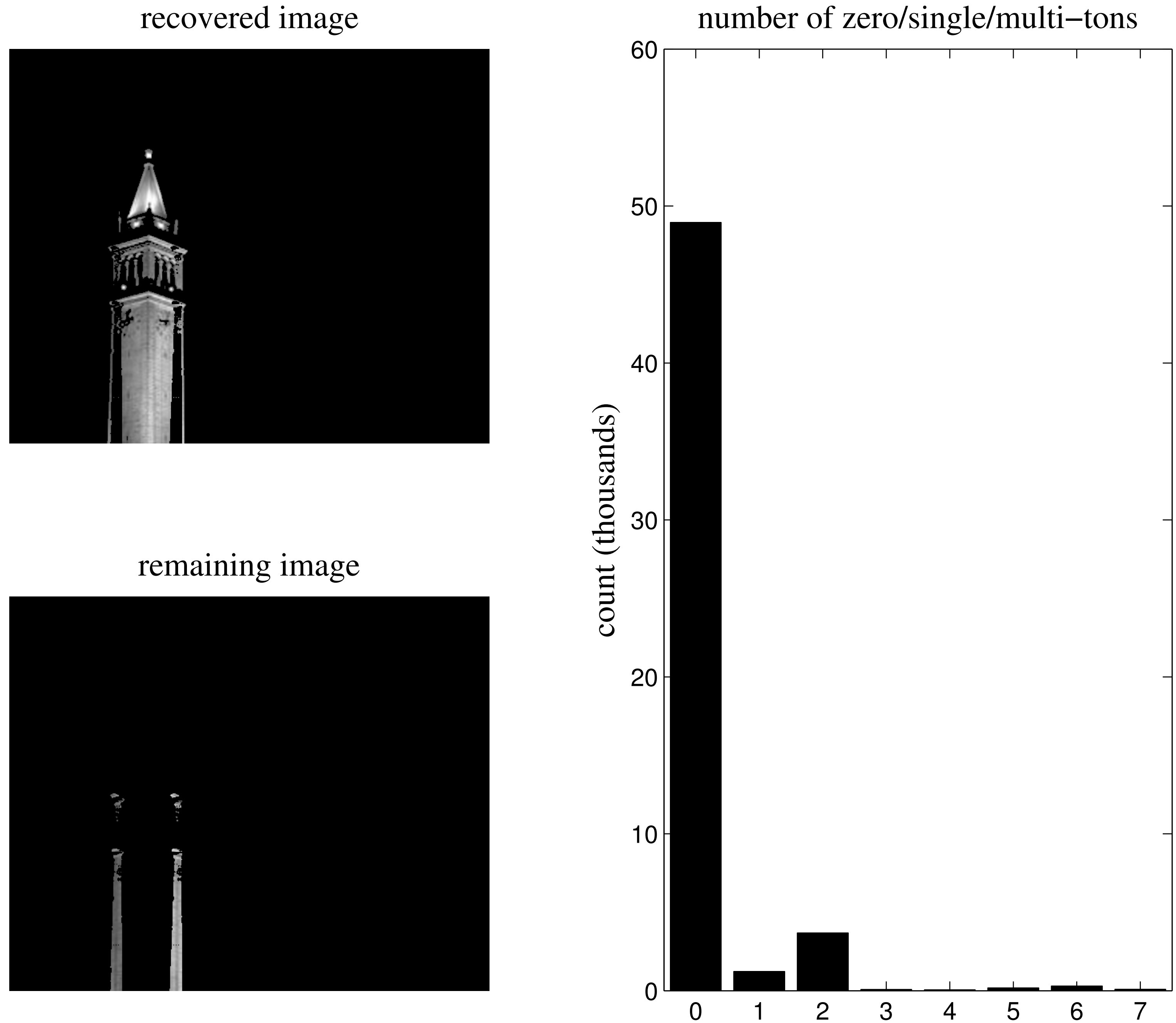
- $N = 600 \times 493 \approx 300$ (thousand)
- $K \approx 25$ (thousand)
- $M \approx 65$ (thousand)

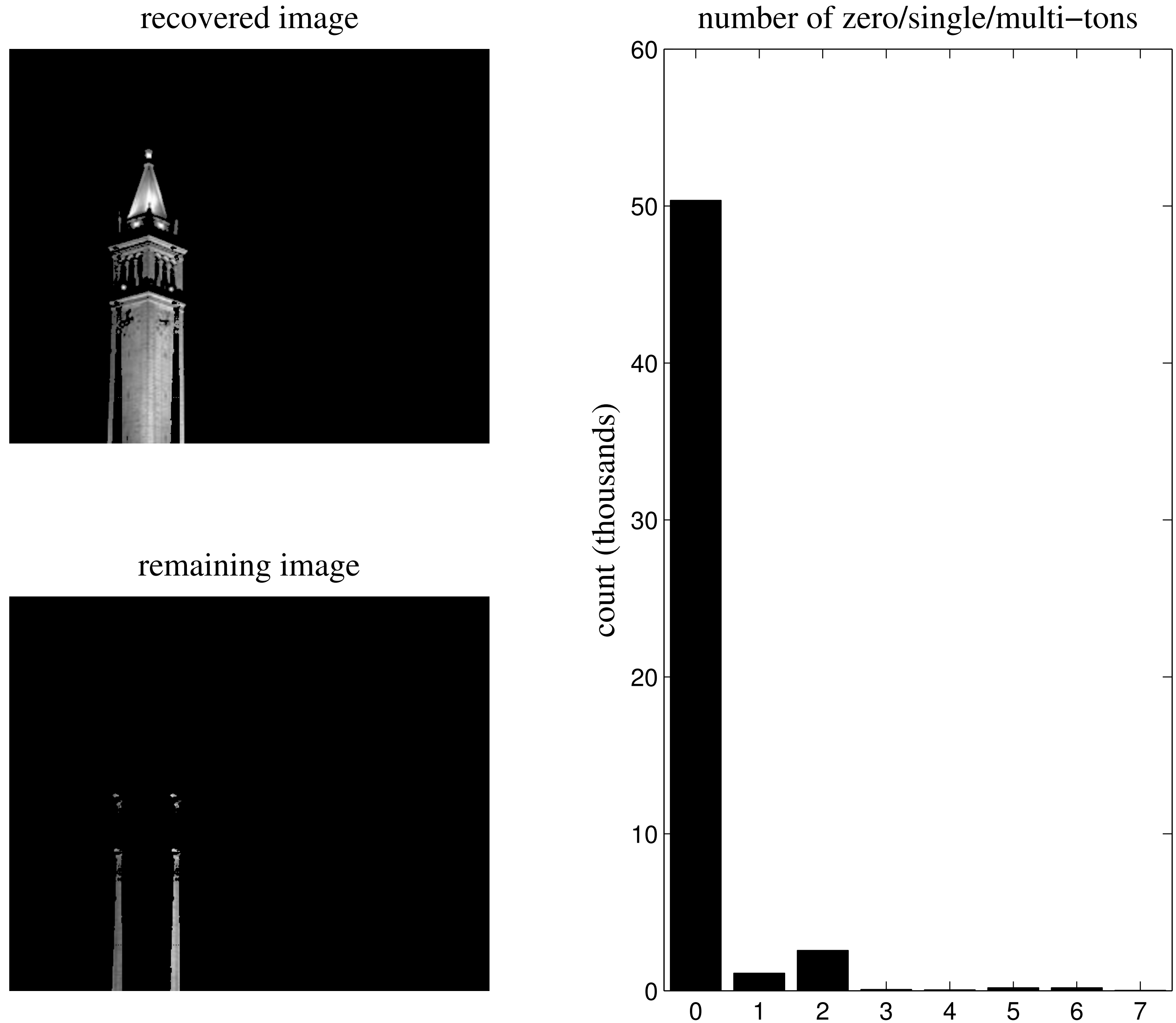

recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7

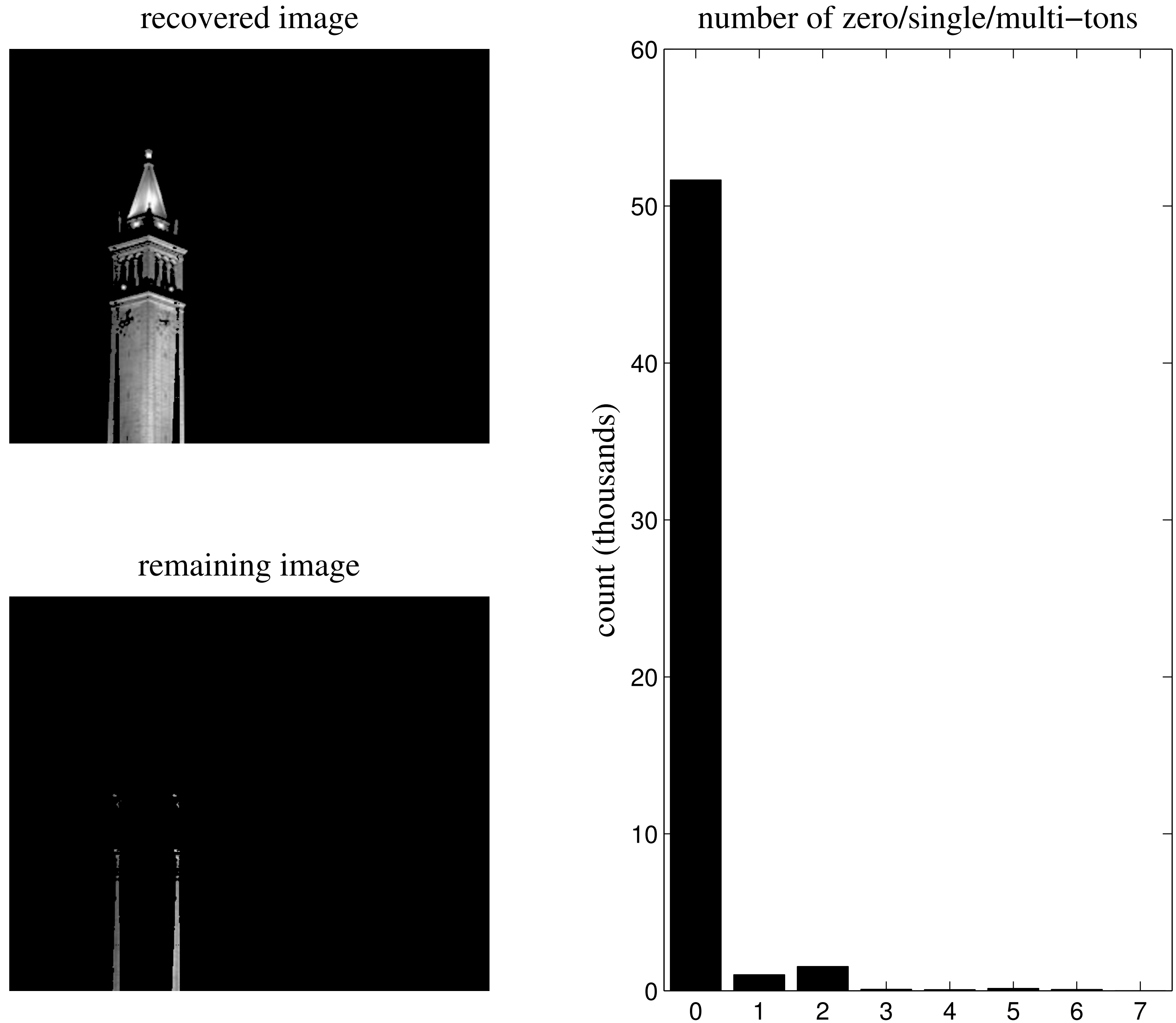

recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7

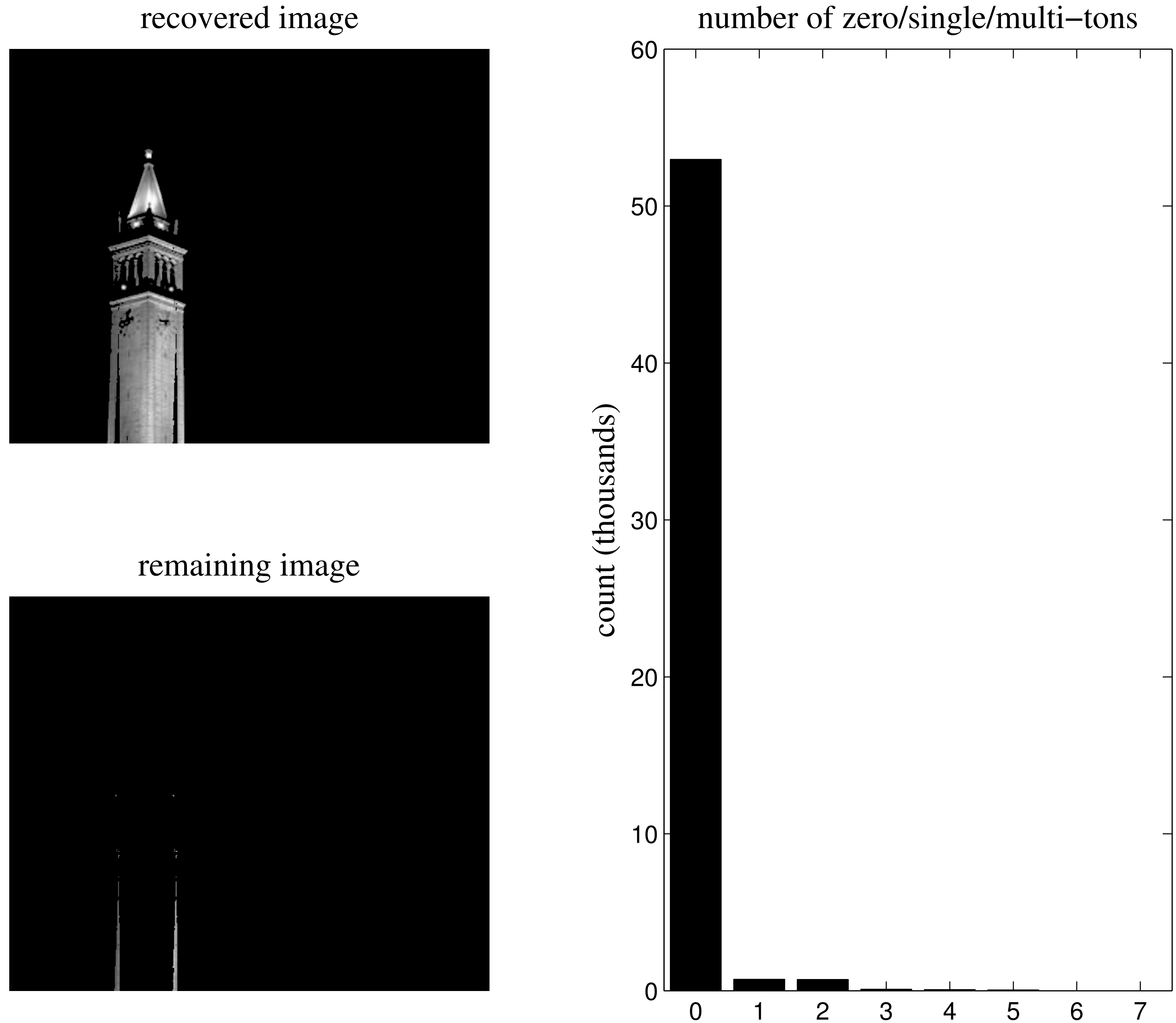


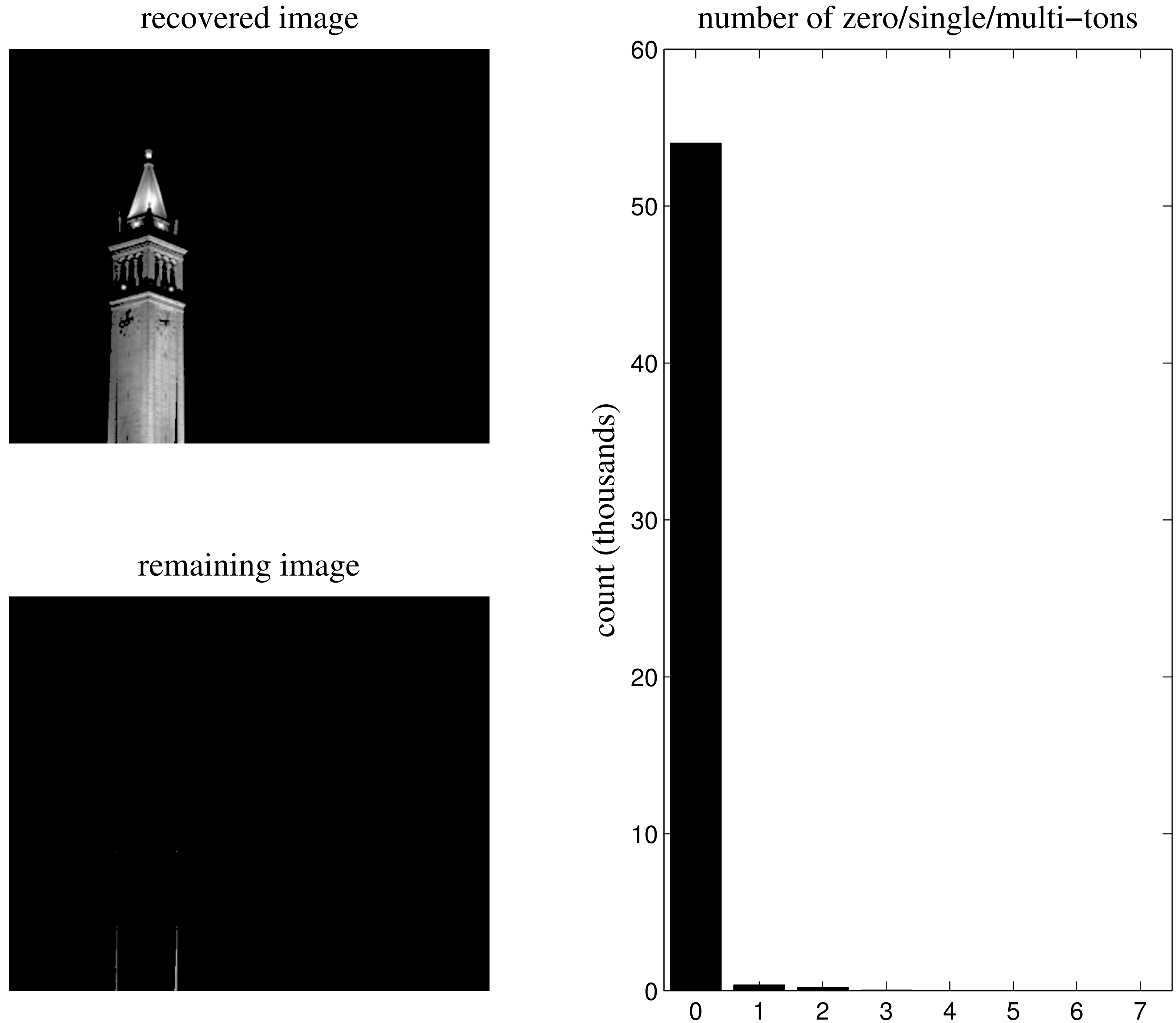
recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0 1 2 3 4 5 6 7

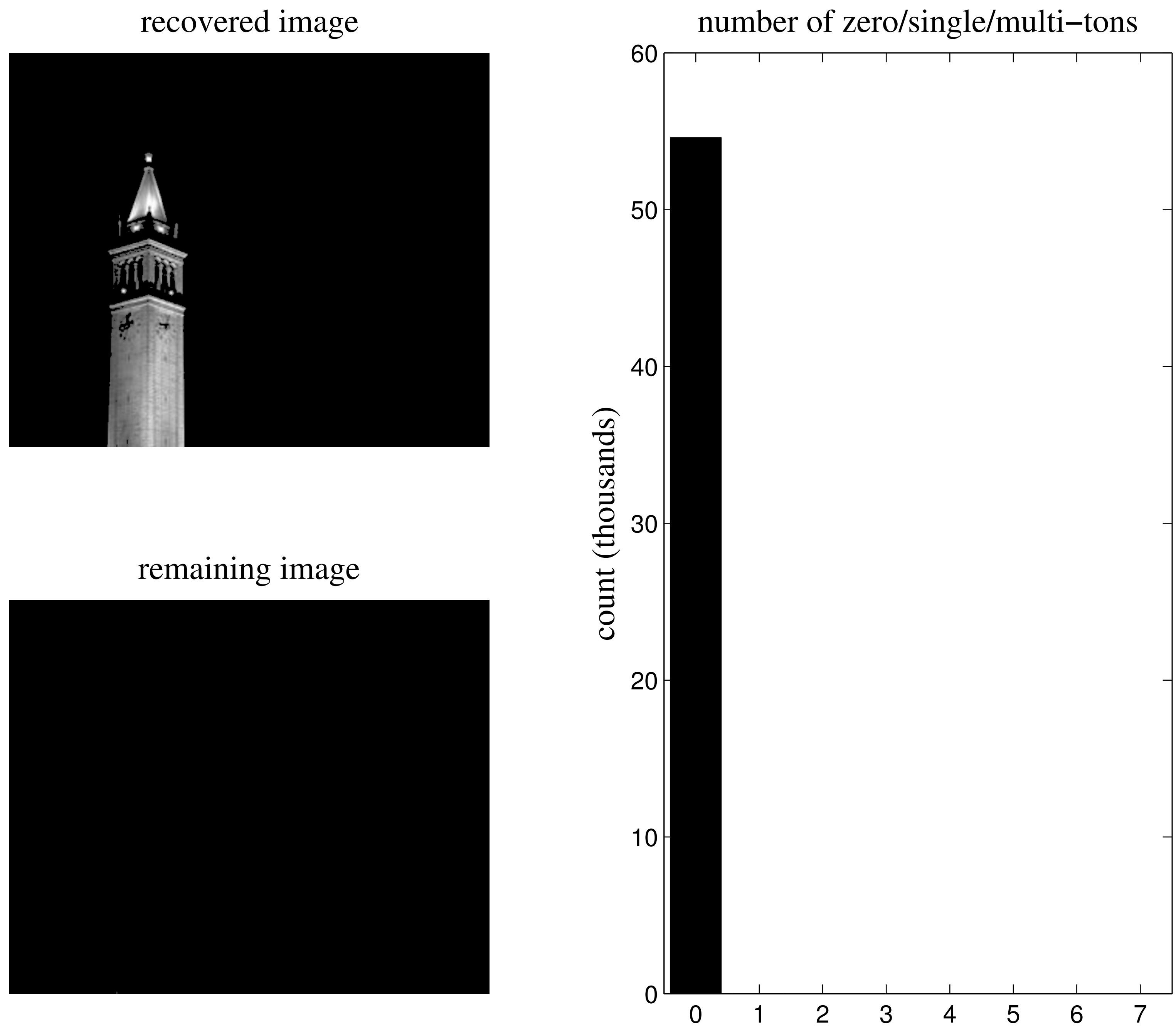

recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7


recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7


recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7

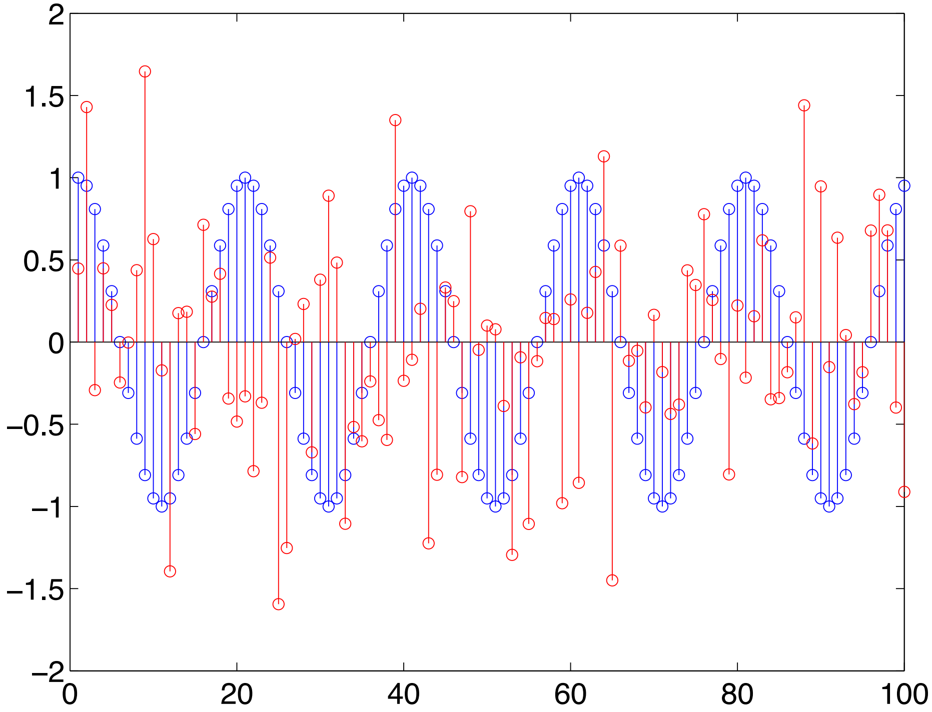

recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
60
50
40
30
20
10
0
0
1
2
3
4
5
6
7

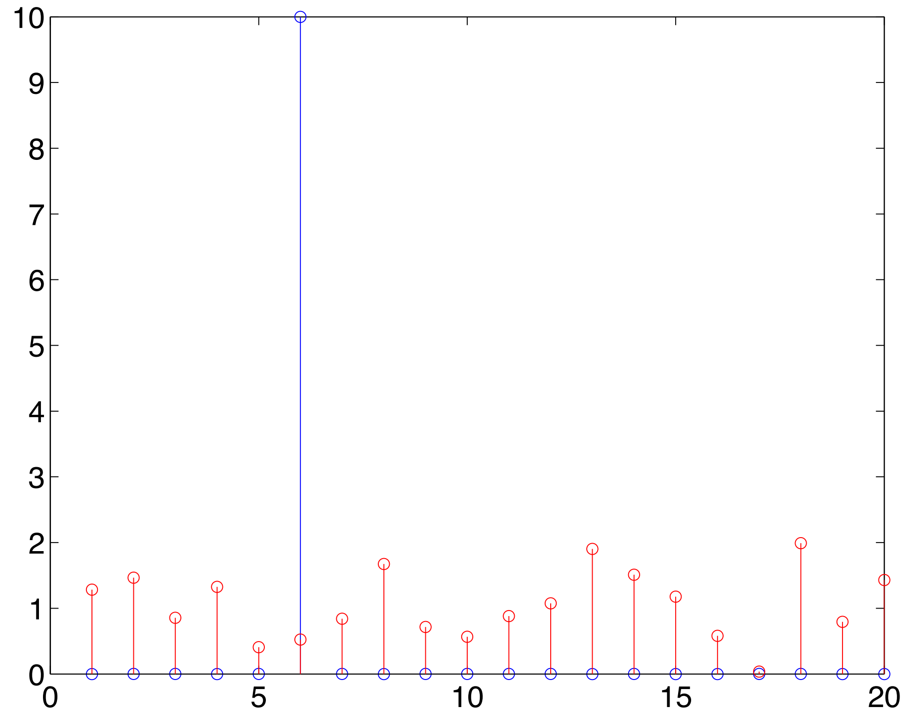

recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7


recovered image
remaining image
number of zero/single/multi−tons
count (thousands)
0
10
20
30
40
50
60
0
1
2
3
4
5
6
7

PART II:
Noisy Recovery

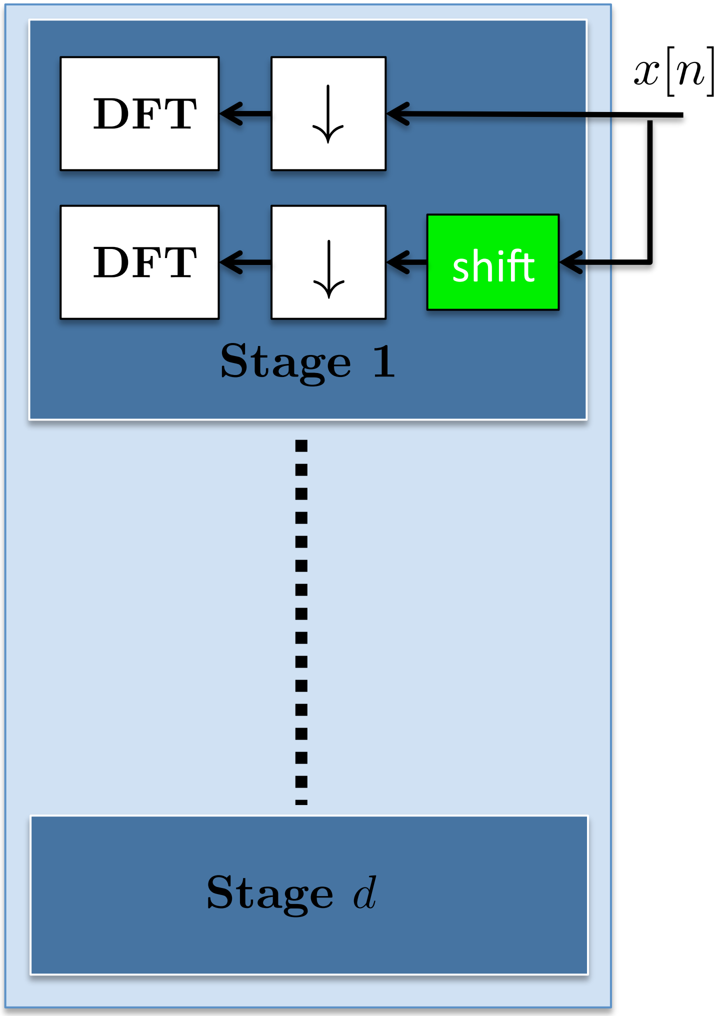
Noisy Setting: R-FFAST | Signal Model



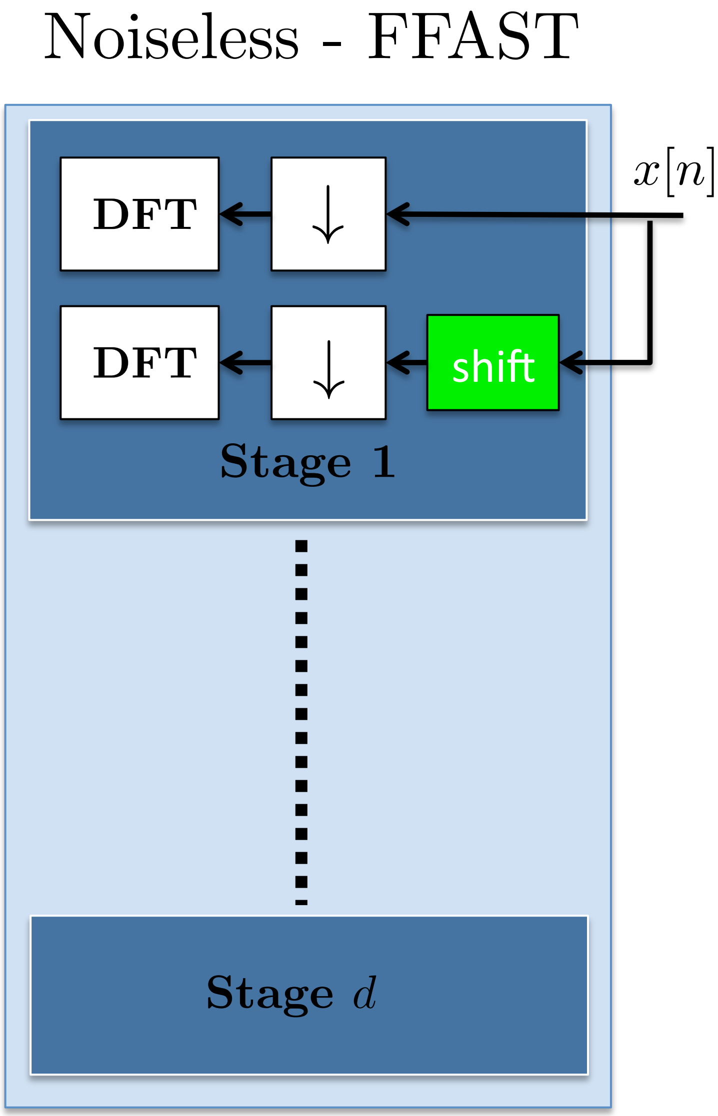


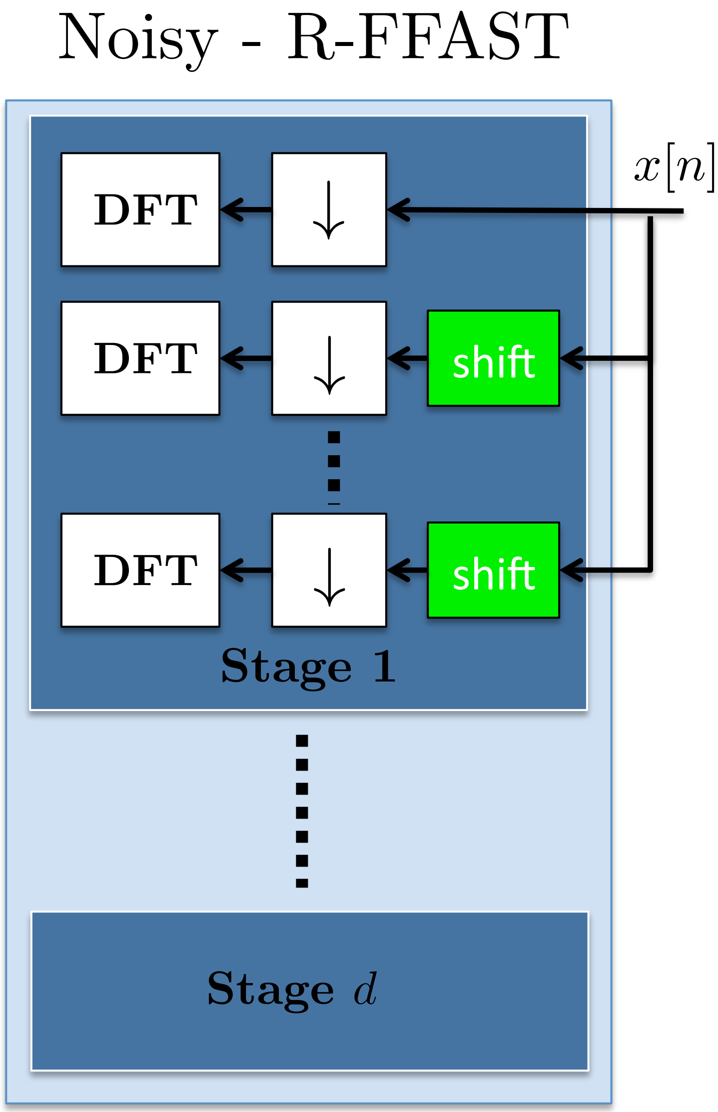
- $y[n] = x[n] + w[n]$, where $w[n] \in \mathcal{CN}(0, \sigma^2)$.
- There are K non-zero $X[k]$, where $X[k]$ is from a finite constellation
- SNR is $\mathbb{E}|x[n]|^2 / \mathbb{E}|w[n]|^2$ (e.g., SNR=0 dB)

Noiseless - FFAST



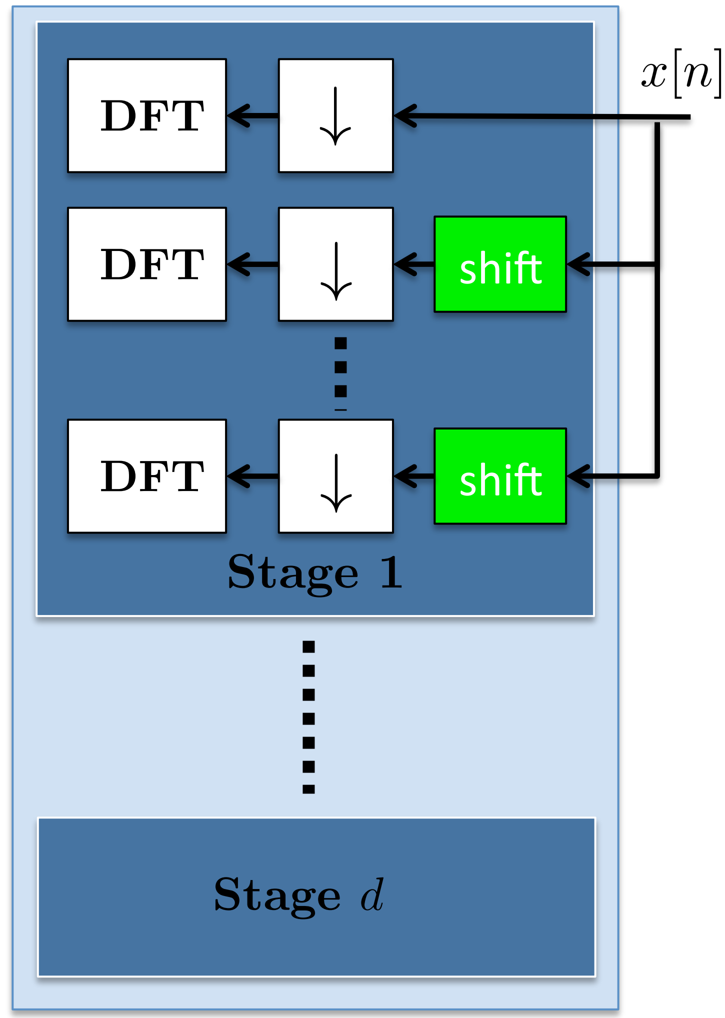
Noisy Setting: R-FFAST | From Noiseless to Noisy





Noisy - R-FFAST

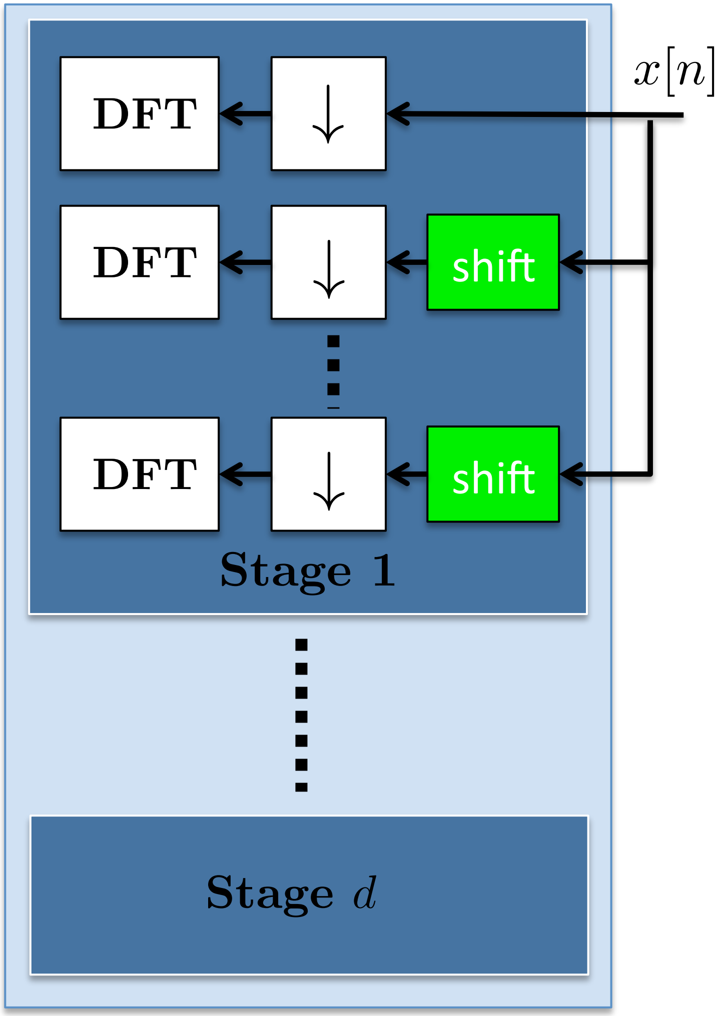
Two schemes to choose **shifts**:



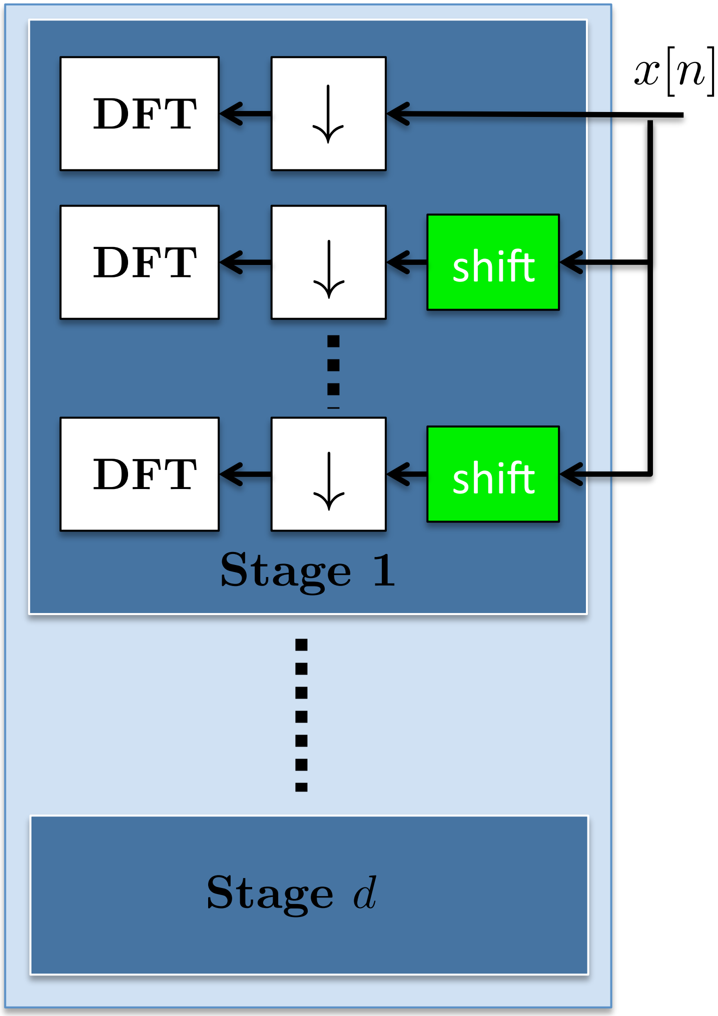
Two schemes to choose **shifts**:

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

Noisy - R-FFAST



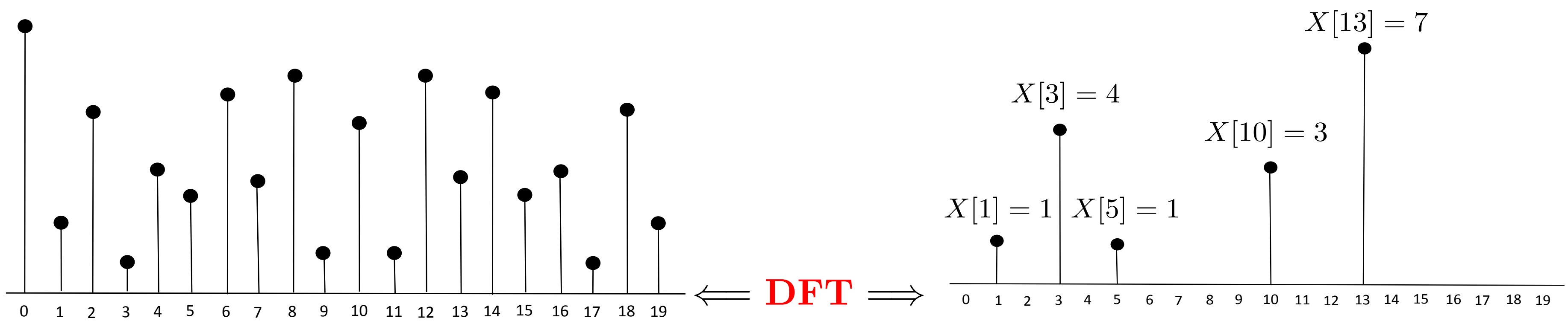
Noisy - R-FFAST

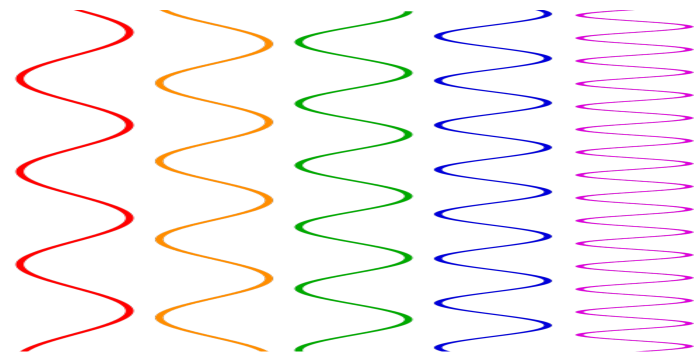


Two schemes to choose **shifts**:

- 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 $M = O(K \log^{1.\dot{3}} N)$
  - **sub-linear** run-time $M = O(K \log^{2.\dot{3}} N)$

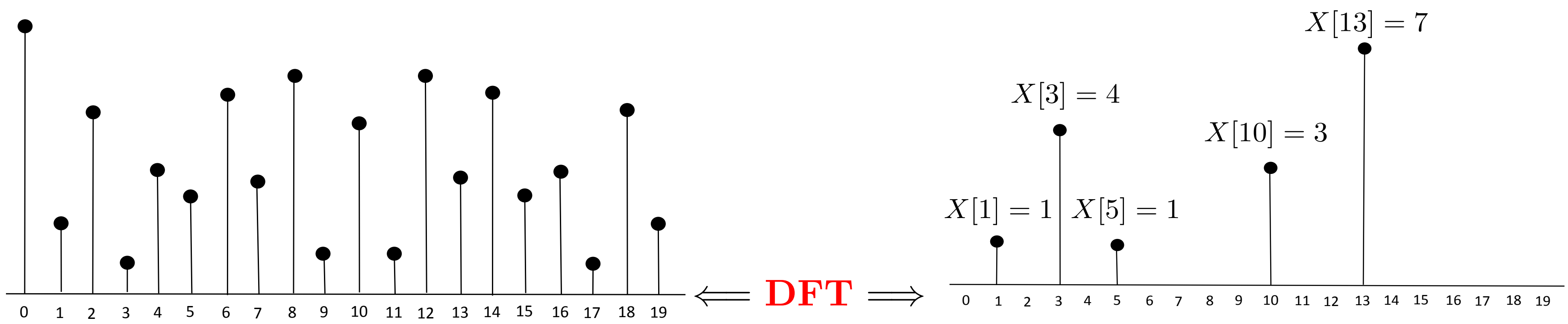
Architecture | Recap on FFAST Sampling





each DFT coefficient
represents a distinct **frequency**

Architecture | Recap on FFAST Sampling



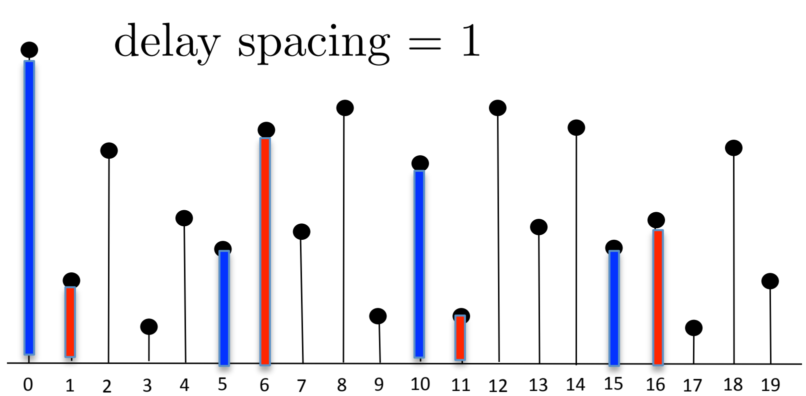
- The key is to pinpoint **single-tons** in terms of
  - location
  - value
- Equivalent to the parameters of a discrete sinusoid
  - frequency
  - amplitude

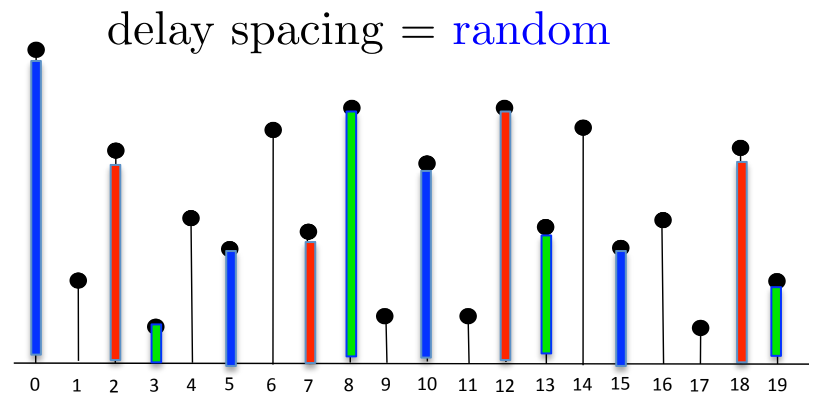
$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

$N \times N$ DFT matrix

$$W = e^{-\mathrm{i}\frac{2\pi}{N}} \text{ with } N = 20$$

Architecture | From Noiseless to Noisy





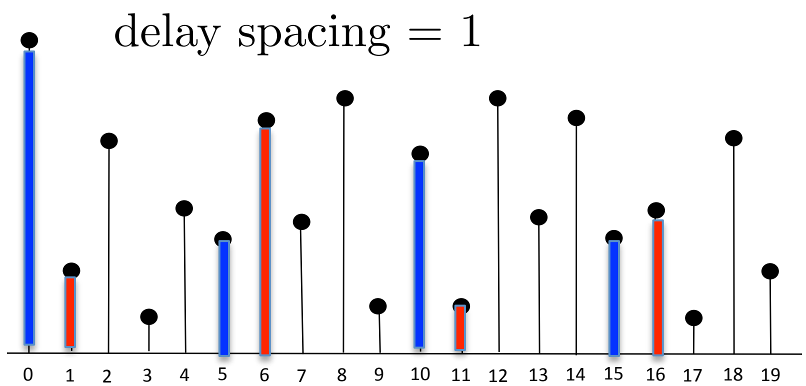
Noiseless

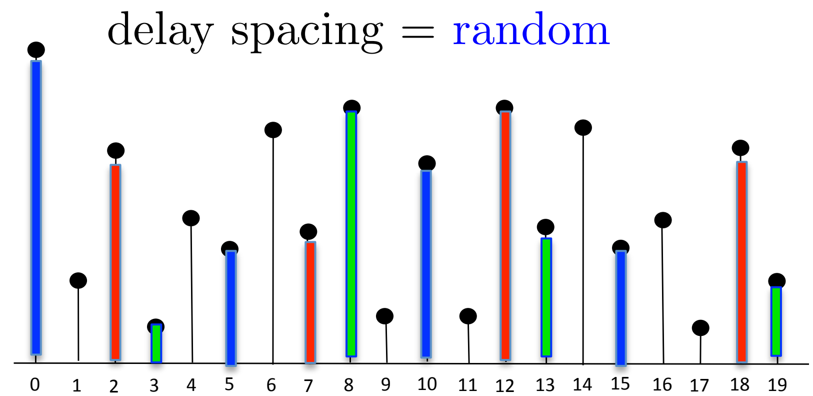
Noisy

matching 2 points of a noiseless sinusoid

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

Architecture | From Noiseless to Noisy





Noiseless

matching 2 points of a noiseless sinusoid

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

Noisy

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

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ 1 & W^3 & W^6 & \cdots & W^{3\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

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

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

- using M noisy samples, where $M = O(K \log N)$, **(order optimal)**
- with probability at least $1 - O(1/M)$,
- in $O(N \log N)$ computations.
- for a finite SNR.

Noisy

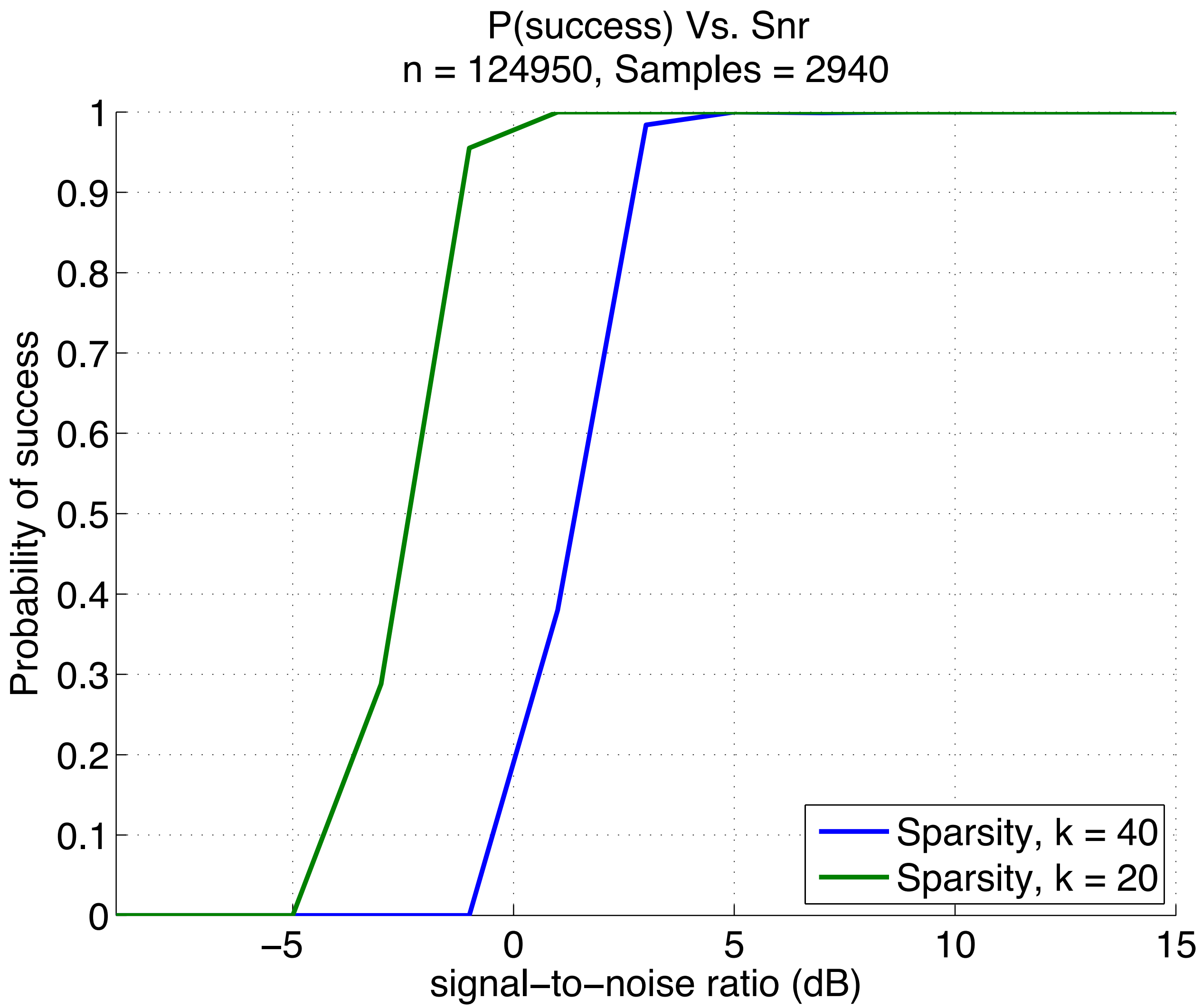
Theorem: FFAST algorithm computes the K-sparse DFT of $\boldsymbol{x} \in \mathbb{C}^N$,

- using $M = O(K)$ samples,
- in $O(K \log K)$ computations,
- with probability at least $1 - O(1/M)$.

Noiseless

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

- $K = 20$ and 40
- $N = 0.124$ million
- Sample cost $M = 2940$



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

- 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

matching 2 points of a noiseless sinusoid

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

Noisy

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

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ 1 & W^3 & W^6 & \cdots & W^{3\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

?

Noisy

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

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ 1 & W^3 & W^6 & \cdots & W^{3\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

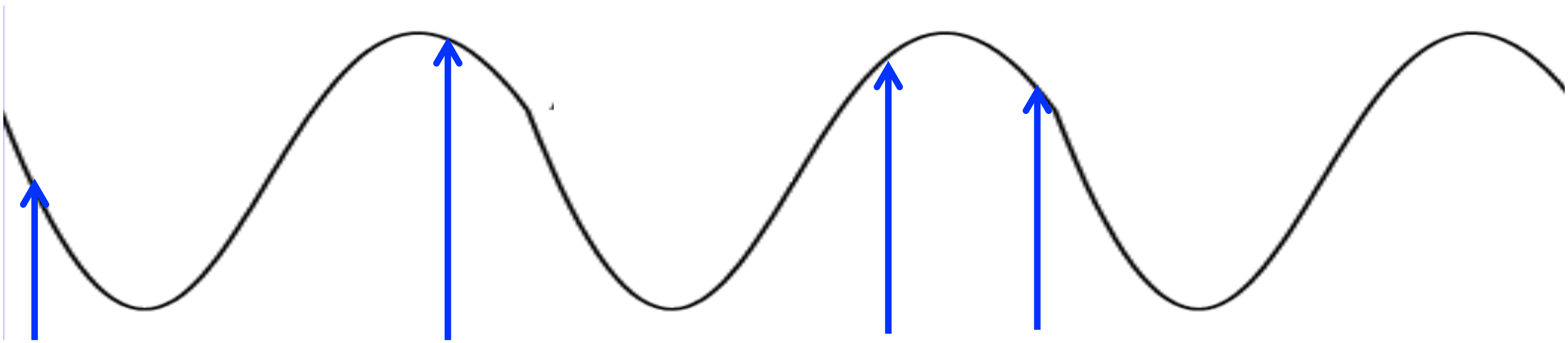
Noisy (sub-linear)

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

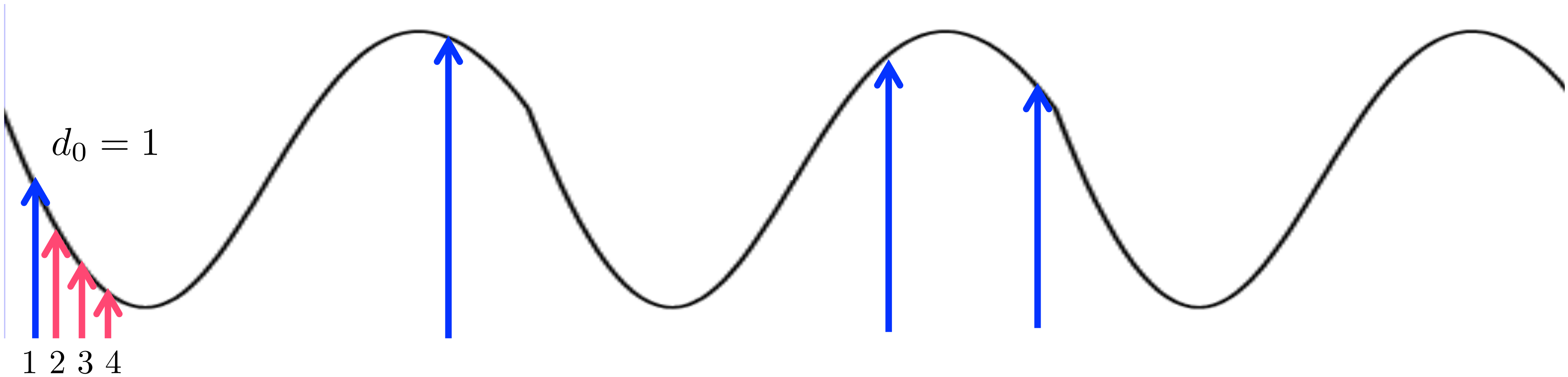
$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{19} \\ 1 & W^2 & W^4 & \cdots & W^{2\times 19} \\ 1 & W^3 & W^6 & \cdots & W^{3\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^k & W^{2k} & \cdots & W^{k\times 19} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & W^{19} & W^{19\times 2} & \cdots & W^{19\times 19} \end{bmatrix}$$

each piece is of length $O(\log^{0.\dot{3}} N)$

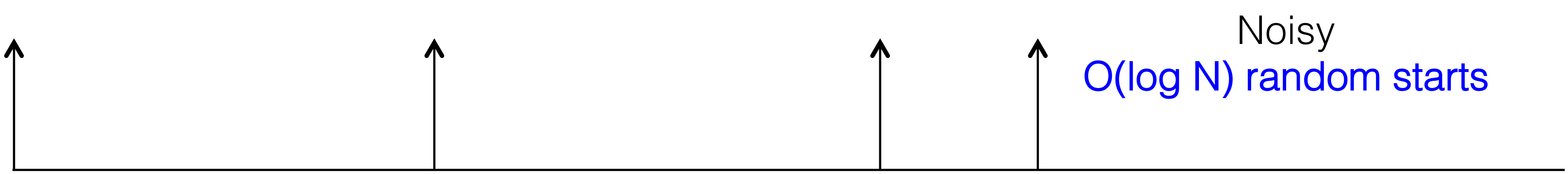
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



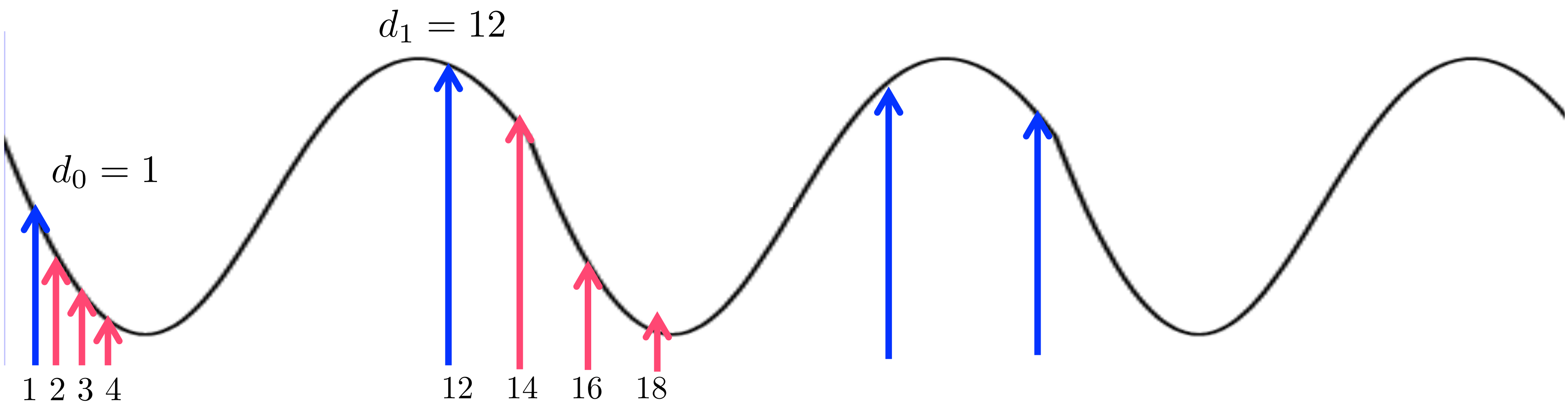
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



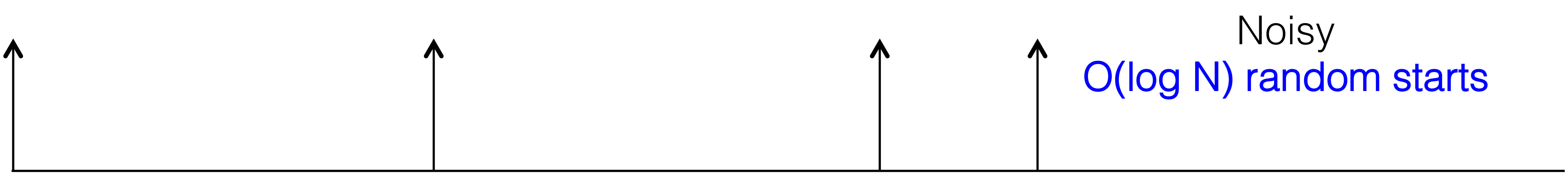
- For each random start, take consecutive delays spaced by 2^i



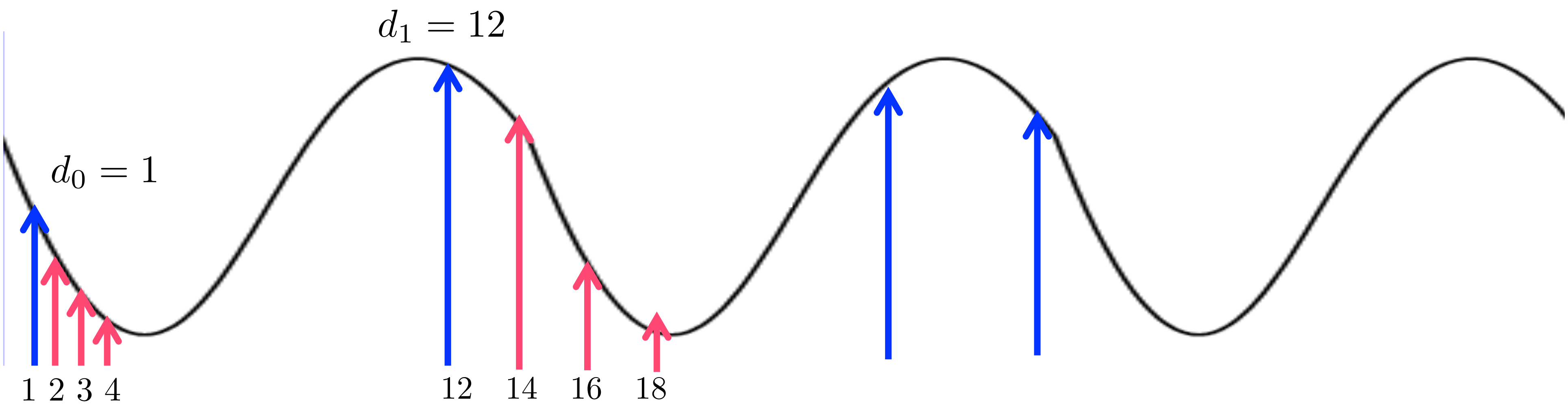
Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



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

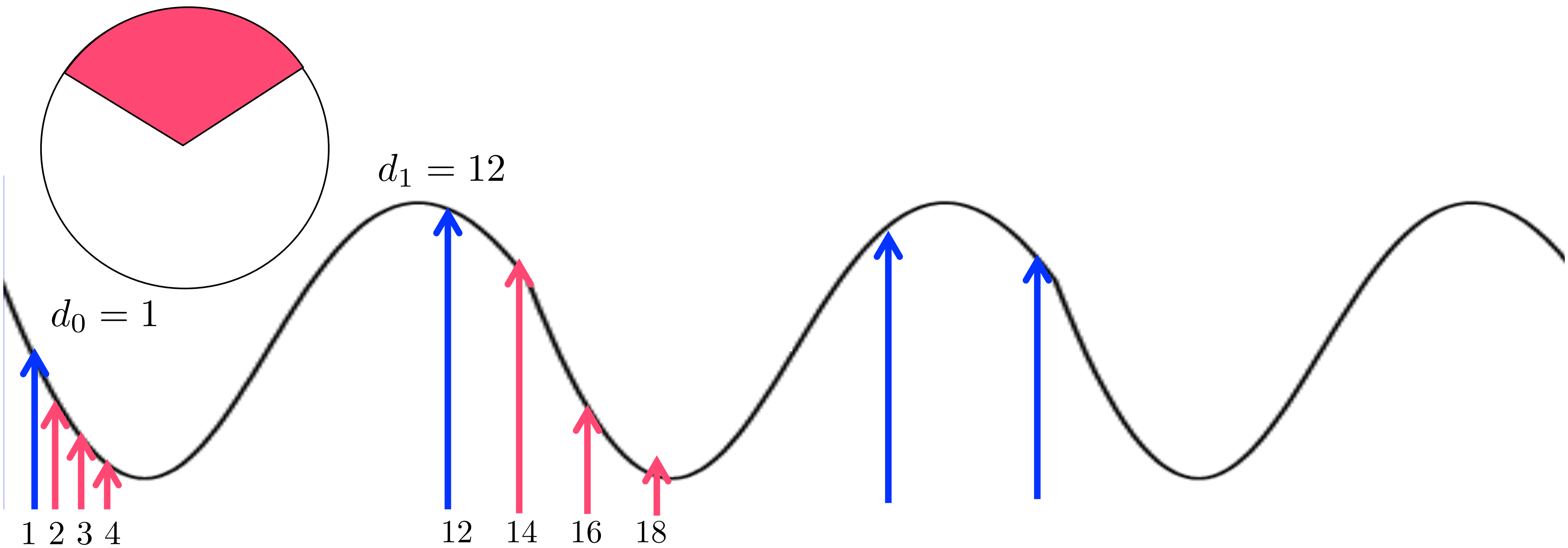


Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



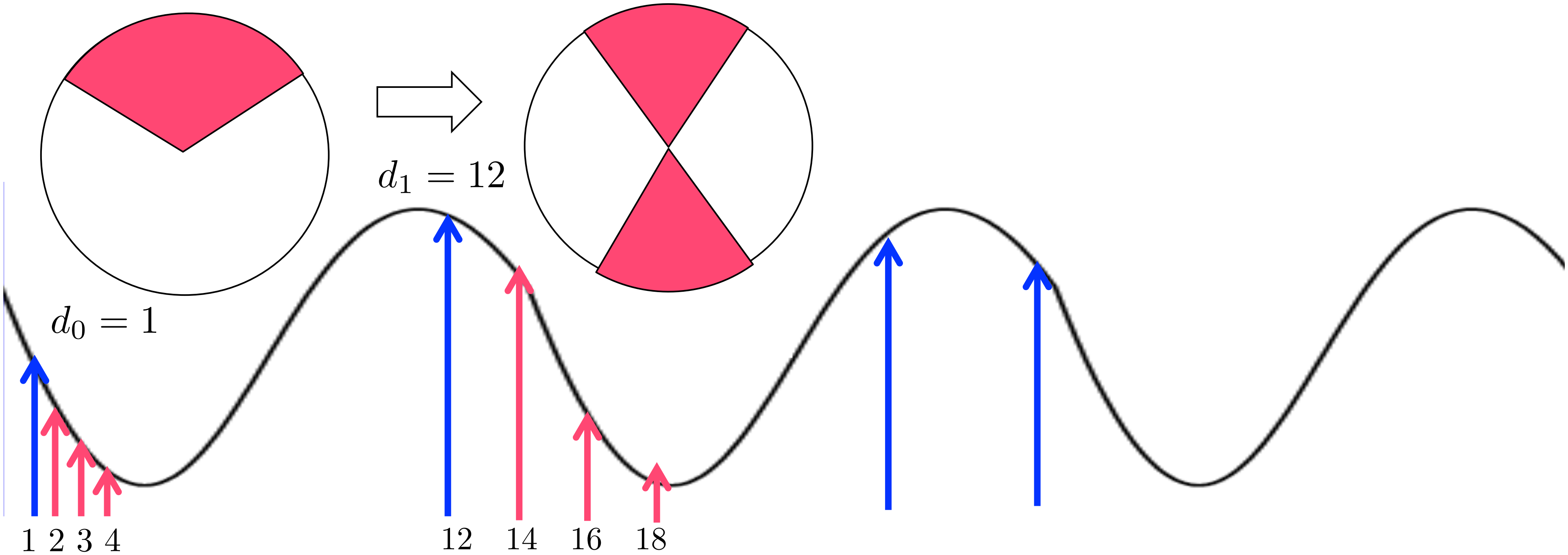
- For each random start, take consecutive delays spaced by 2^i
- [1989'Kay] provides an unbiased and efficient estimate of $2^i\omega$
- $O(\log^{1/3} N)$ consecutive rows are **sufficient** for matching each piece

Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



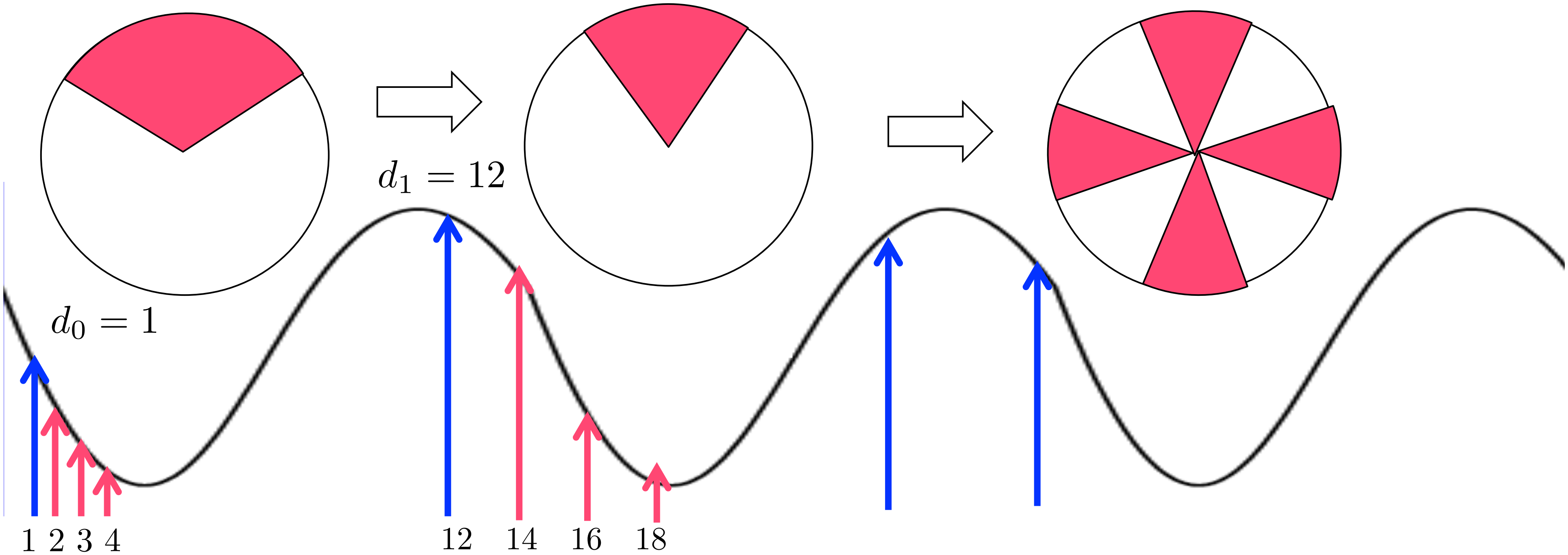
- **Piece 1**: estimates ω with no ambiguity

Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



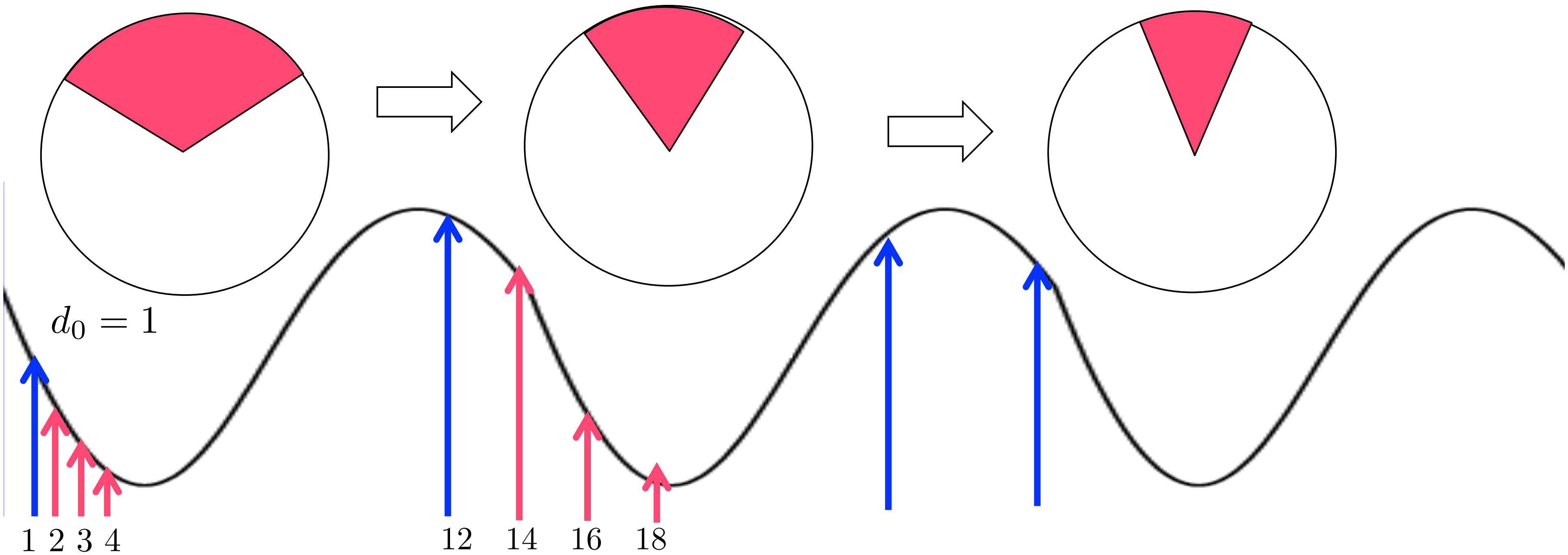
- **Piece 1**: estimates ω with no ambiguity
- **Piece 2**: estimates $2\omega \implies$ unwrapping leads to 2 ambiguities

Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time



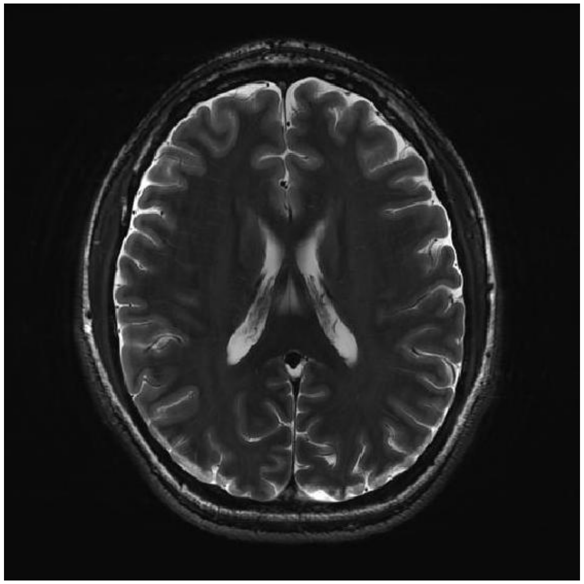
- **Piece 1**: estimates ω with no ambiguity
- **Piece 2**: estimates $2\omega \implies$ unwrapping leads to 2 ambiguities
- **Piece i**: estimates $2^i\omega \implies$ unwrapping leads to 2^i ambiguities

Noisy Setting: R-FFAST | Near Sample-Optimal Recovery with Sub-linear Time

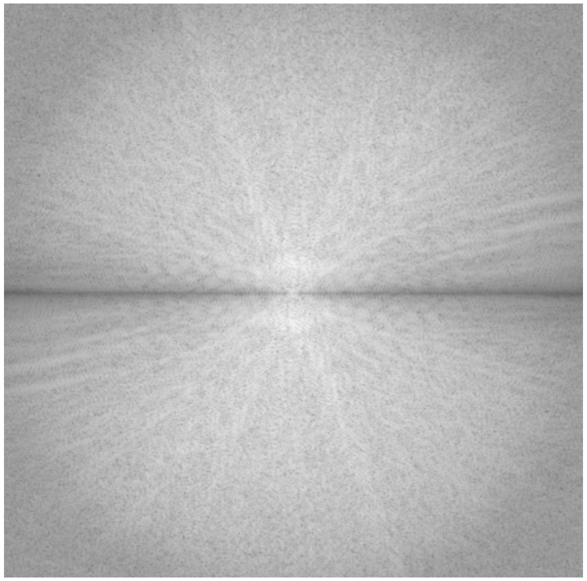


- accuracy improves dyadically 2^i
- each piece provides 1 information bit
- $\log N$ pieces $\Longrightarrow$ N locations

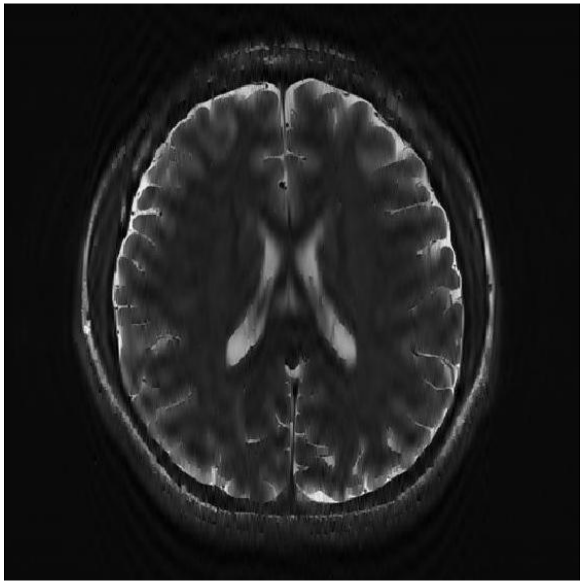
Computing Sparse DFT | Some Concrete Examples of FFAST



Original Brain image

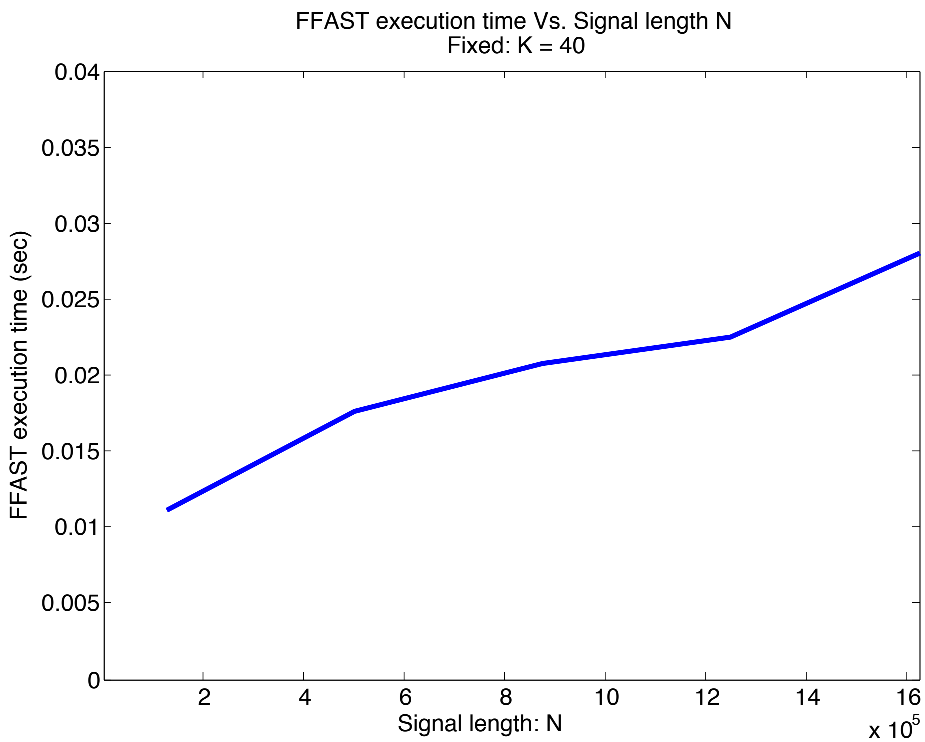


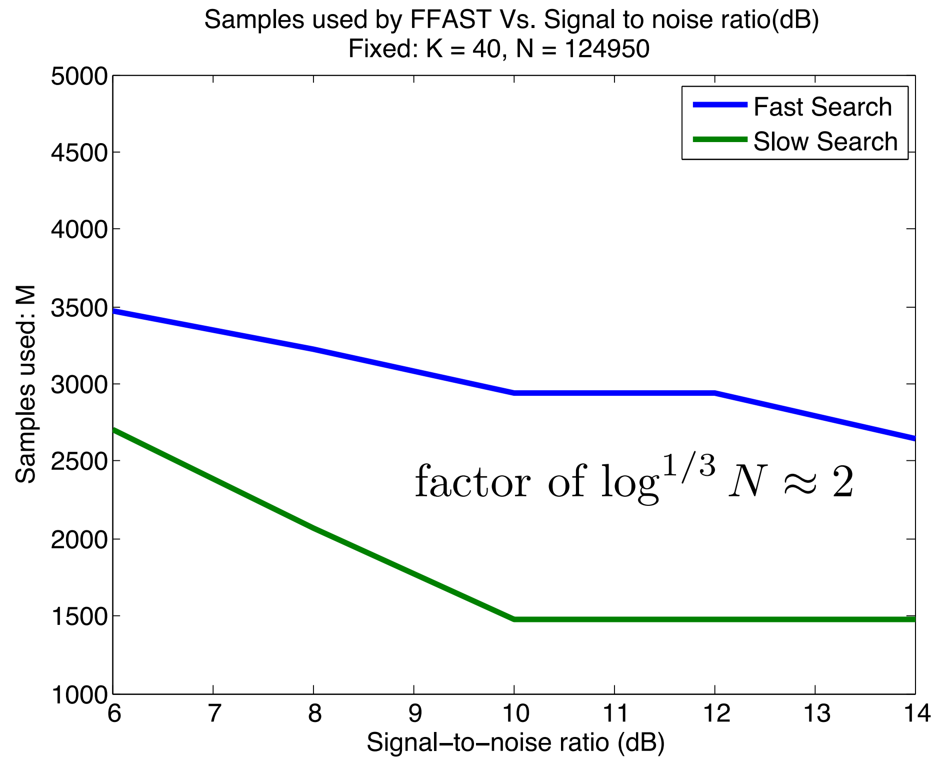
Fourier domain of
the Brain Image



Brain image
reconstructed by FFAST

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 $\{\pm 1, \pm j\}$, 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).

- **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*]
  Xiao Li; Bradley, J.K.; Pawar, S.; Ramchandran, K., "The SPRIGHT algorithm for robust sparse Hadamard Transforms," Information Theory (ISIT), 2014 IEEE International Symposium on , vol., no., pp.1857,1861, June 29 2014-July 4 2014.

Sparse Iterative Graph-based Hadamard Transform (SPRIGHT)

Conclusion

C++ code available!

- 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_ffft/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 $\Longrightarrow$ **“PhaseCode”** design
    http://arxiv.org/abs/1408.0034