EE 123 Discussion Section 6

Frank Ong

March 14th, 2016

Plan

- Sparse FFT
- Magnitude Filter Design with convex optimization

Sparse FFT

- Given a length-N signal,
- FFT takes O(N log N) time to compute its DFT

- What if our DFT is k-sparse, ie, has only k non-zero entries,
- How much faster can we compute the DFT?

Sparse FFT

- Given a length-N signal,
- FFT takes O(N log N) time to compute its DFT

- What if our DFT is k-sparse, ie, has only k non-zero entries,
- How much faster can we compute the DFT?
- O(k log k) if non-zero locations are randomized

The FFAST algorithm: https://www.eecs.berkeley.edu/~kannanr/project_ffft.html

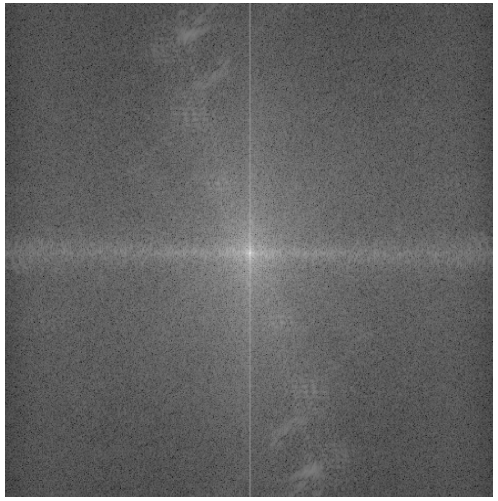
Three Ideas behind FFAST

- Sparse spectrum does not alias much
- 1-sparse DFT is easy
- Different sampling factor gives different aliasing pattern

Uses basic DSP sampling theory

Aliasing with Dense Spectrum

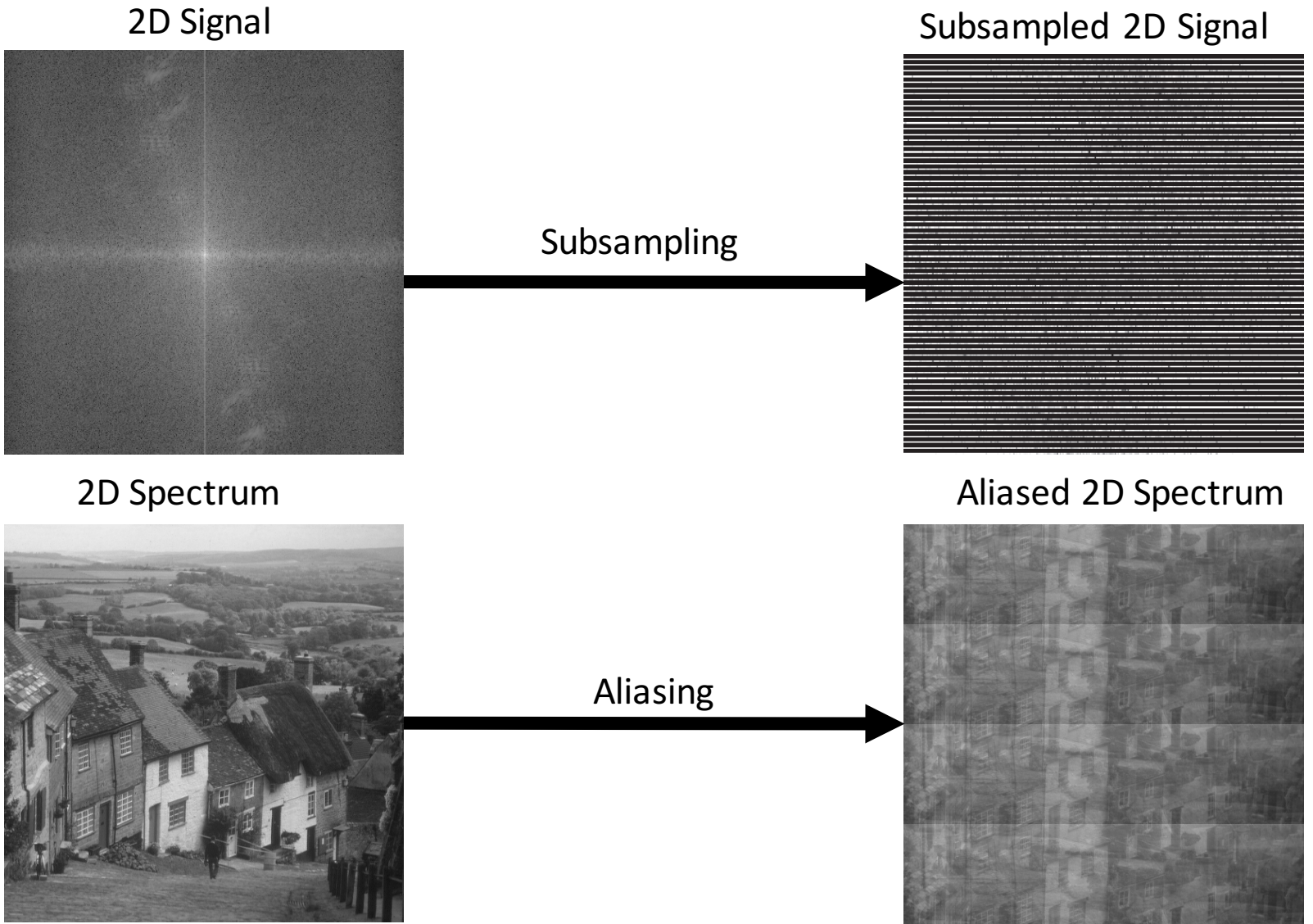
2D Signal



2D Spectrum

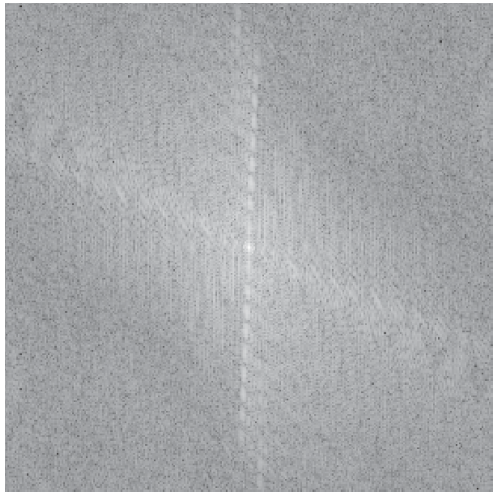
Aliasing with Dense Spectrum

- Everything gets aliased on top of each other



Aliasing with Sparse Spectrum

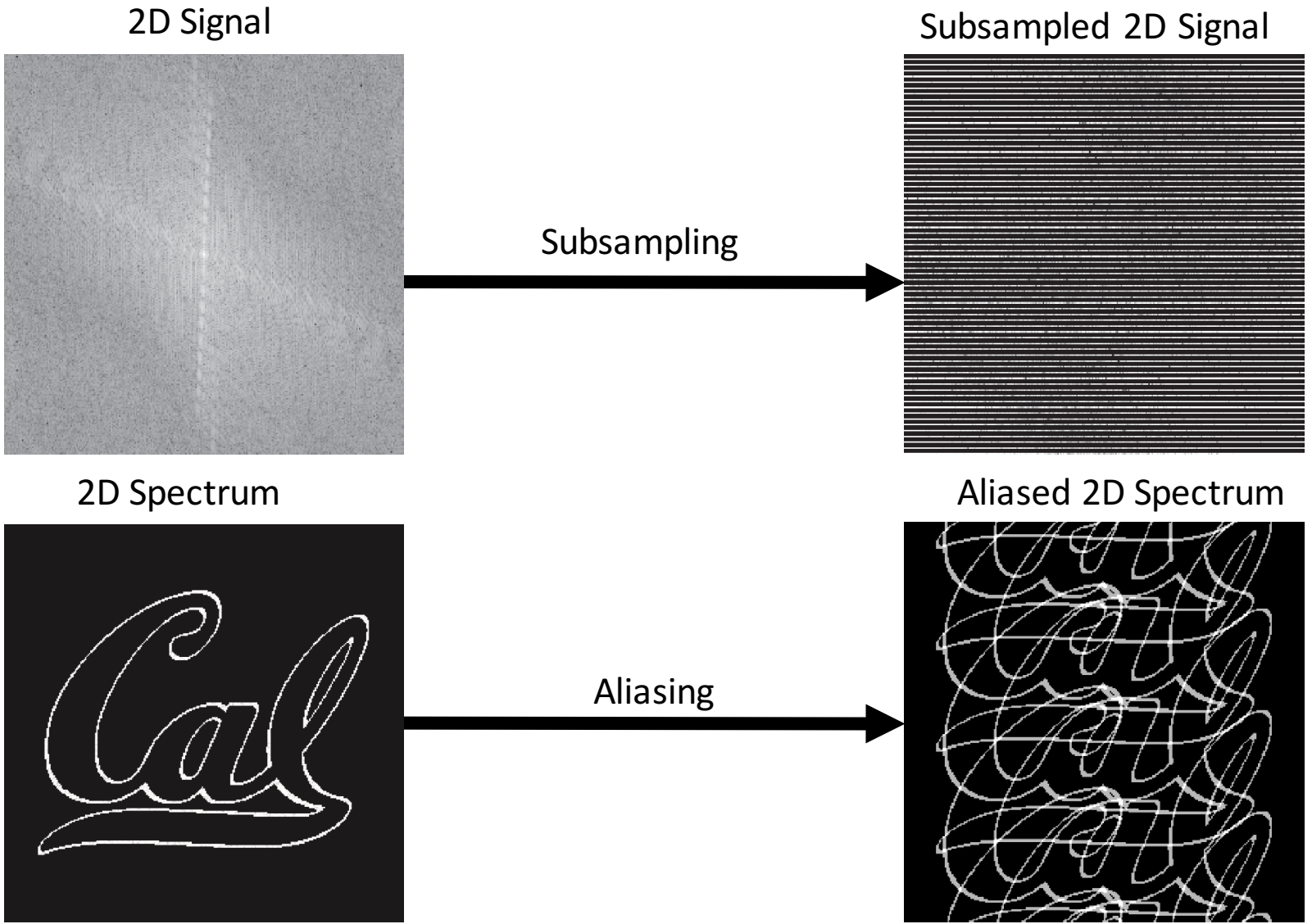
2D Signal



2D Spectrum

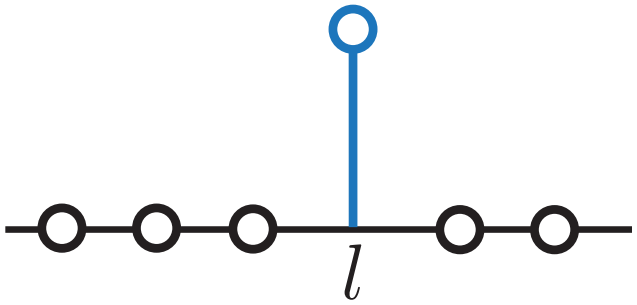
Aliasing with Sparse Spectrum

- Most entries do not have aliasing!



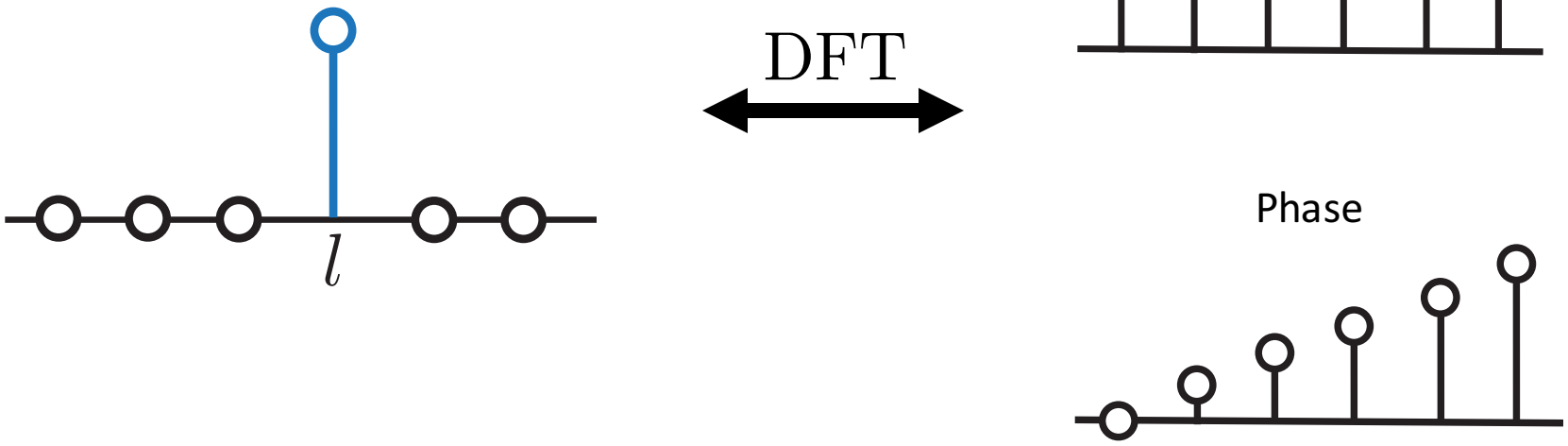
1-Sparse DFT

$$X[k] = \delta[k - l]$$



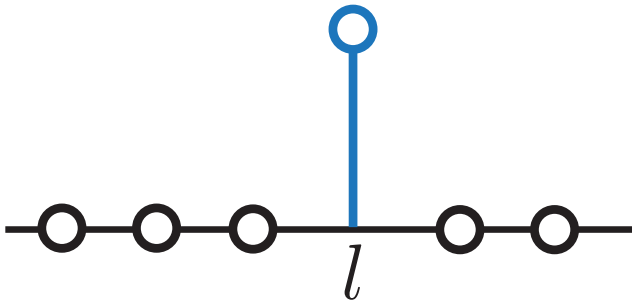
1-Sparse DFT

$$X[k] = \delta[k - l] \qquad x[n] = e^{j\frac{2\pi}{N}ln}$$

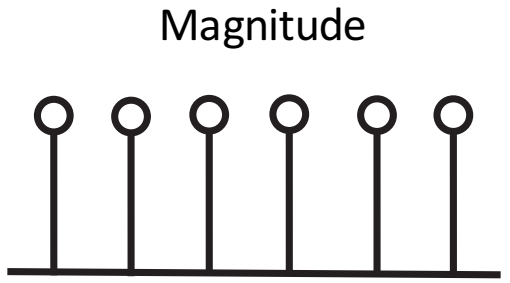


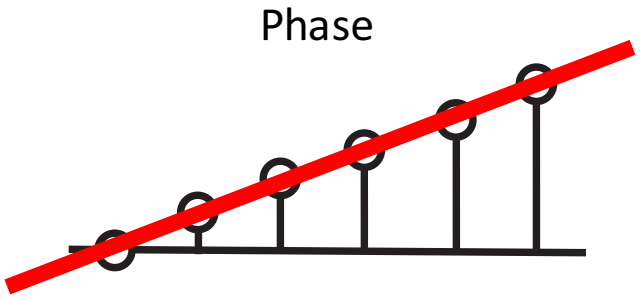
1-Sparse DFT

$$X[k] = \delta[k-l] \qquad x[n] = e^{j\frac{2\pi}{N}ln}$$



DFT





Slope = location

1-Sparse DFT

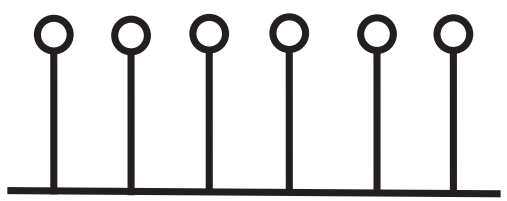
$$X[k] = \delta[k - l] \qquad x[n] = e^{j\frac{2\pi}{N}ln}$$

- For noise-less, needs only 2 samples,

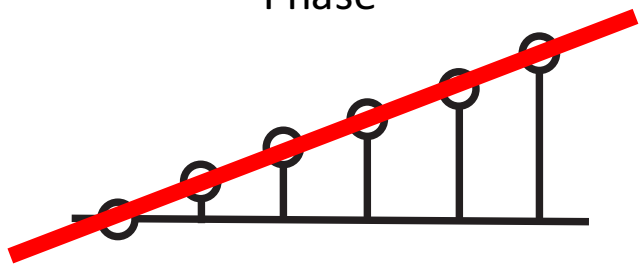
$$\angle(x[1]x^*[0]) = \frac{2\pi}{N}l$$

- Constant computation time

Magnitude

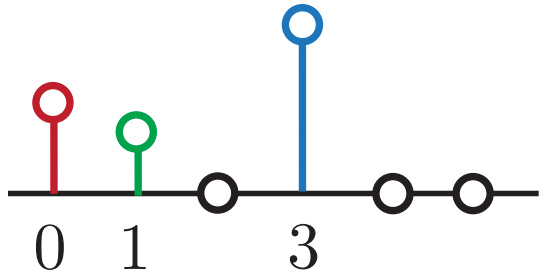


Phase

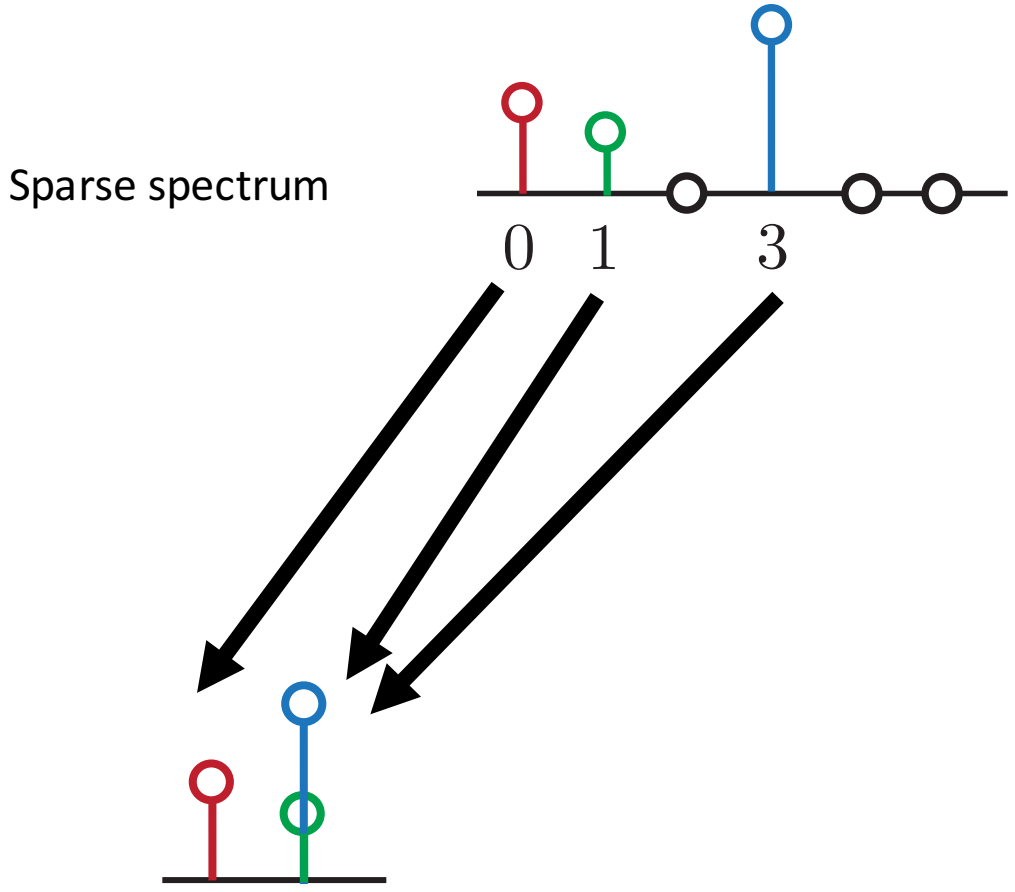


Different subsampling produces different aliasing

Sparse spectrum

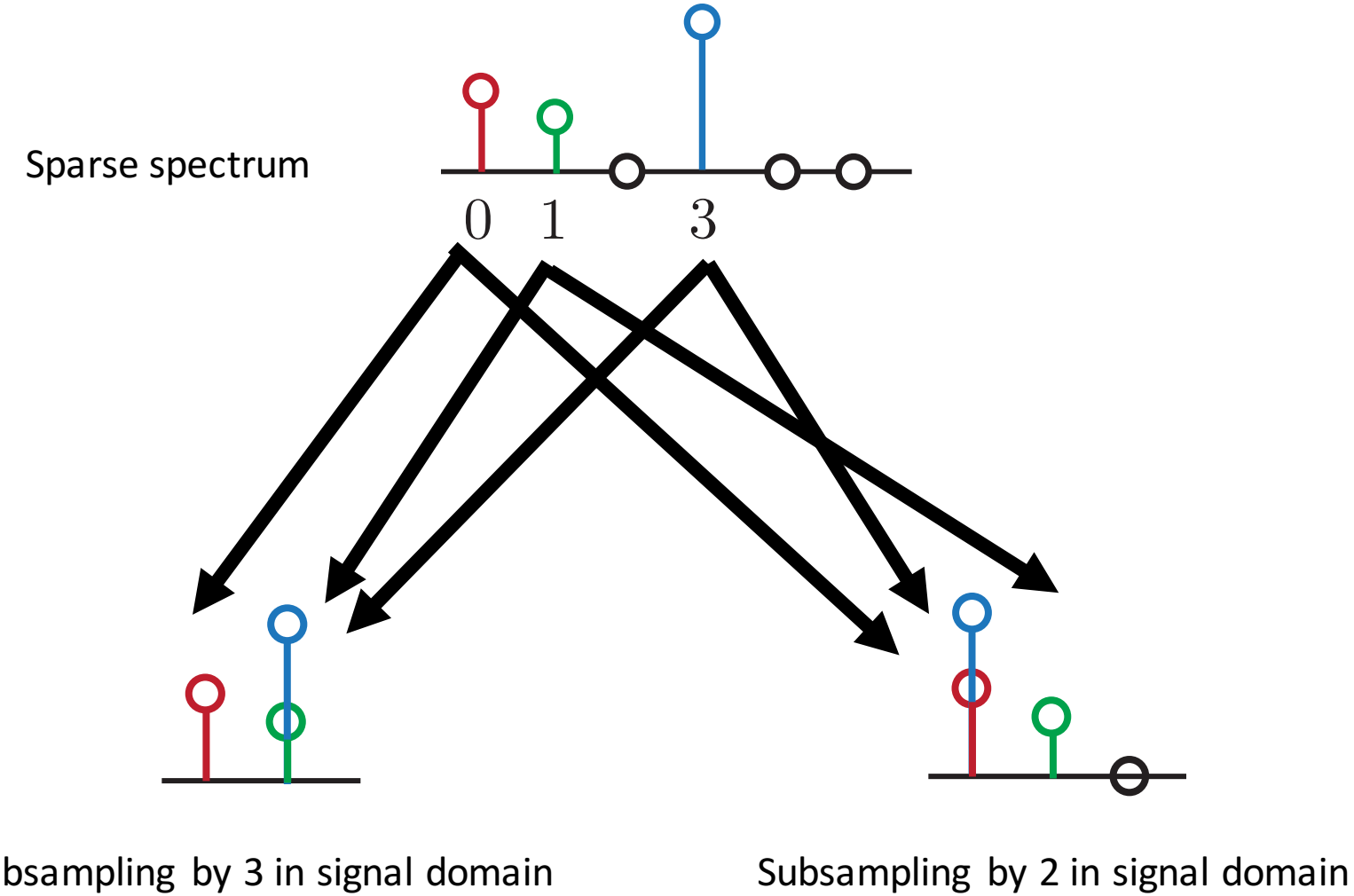


Different subsampling produces different aliasing

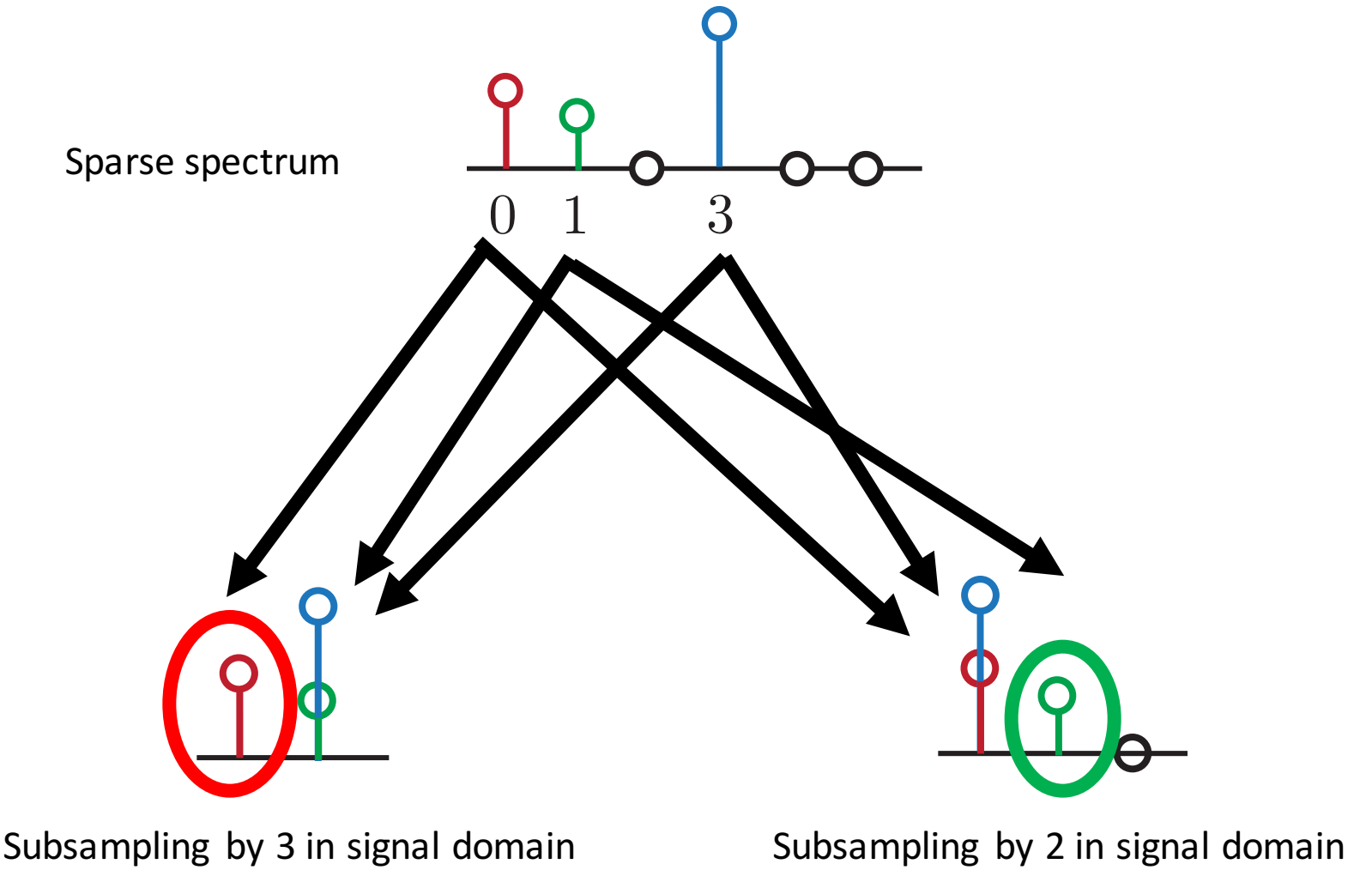


Subsampling by 3 in signal domain

Different subsampling produces different aliasing



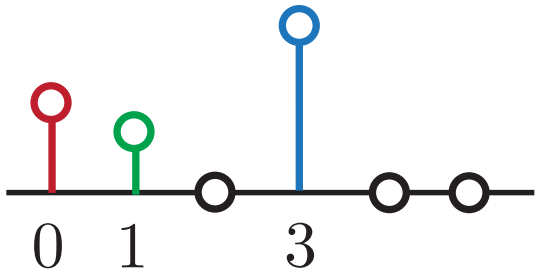
Different subsampling produces different aliasing



- Red and green are not aliased for different sampling factors

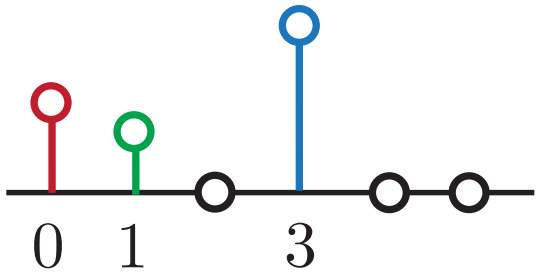
Combining three ideas

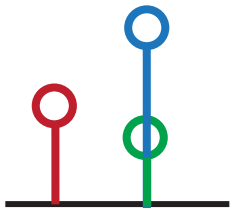
Sparse spectrum



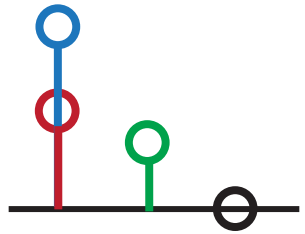
Combining three ideas

Sparse spectrum





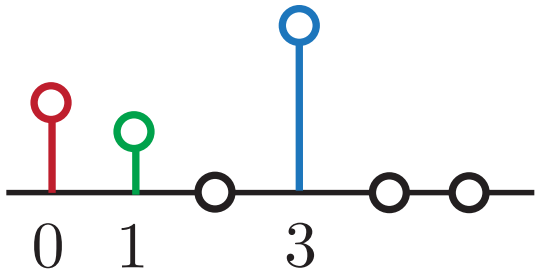
Subsampling by 3 in signal domain

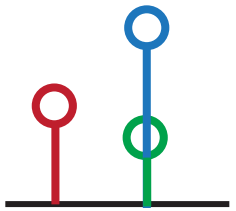


Subsampling by 2 in signal domain

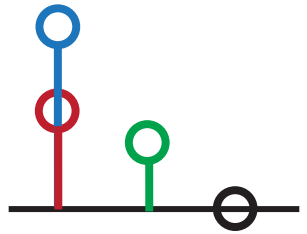
Combining three ideas

Sparse spectrum

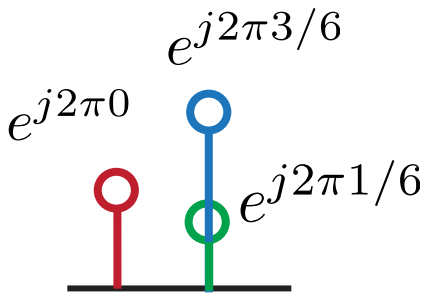




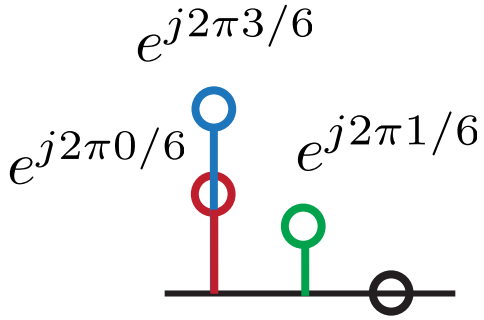
Subsampling by 3 in signal domain



Subsampling by 2 in signal domain



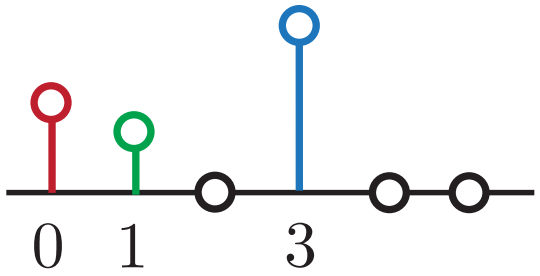
Shift sampling pattern by 1



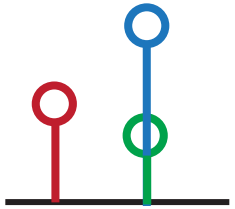
Shift sampling pattern by 1

Combining three ideas

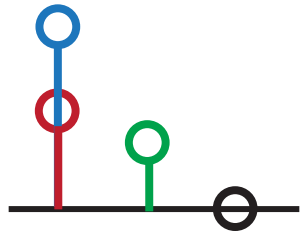
Sparse spectrum



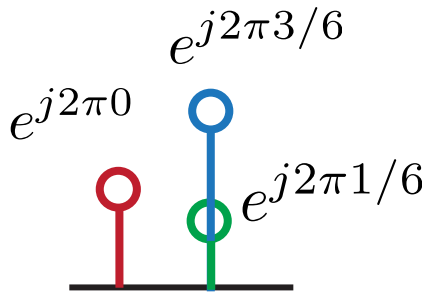
Can recover red and green locations via phase differences



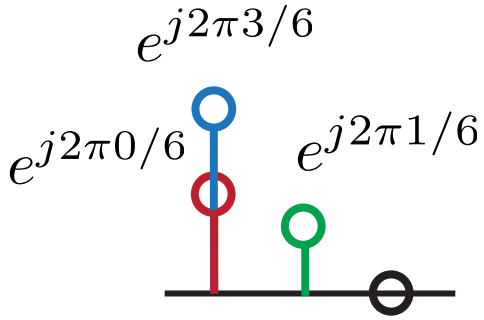
Subsampling by 3 in signal domain



Subsampling by 2 in signal domain



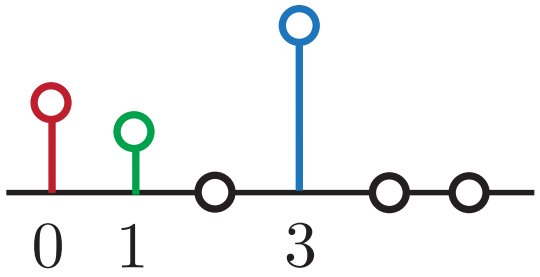
Shift sampling pattern by 1



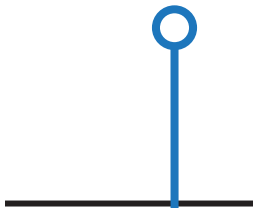
Shift sampling pattern by 1

Combining three ideas

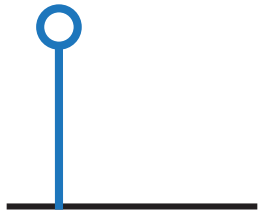
Sparse spectrum



Can peel off red and green

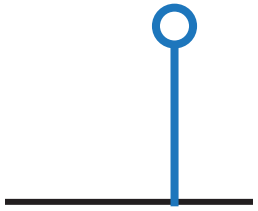


Subsampling by 3 in signal domain



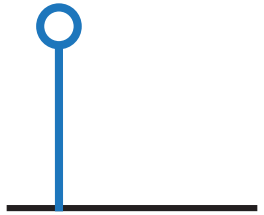
Subsampling by 2 in signal domain

$e^{j2\pi 3/6}$



Shift sampling pattern by 1

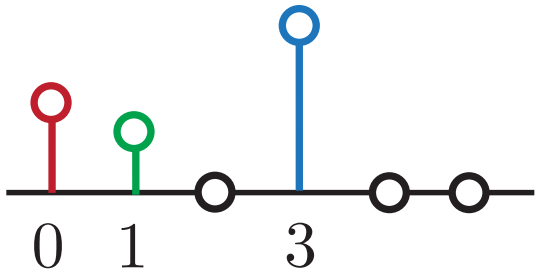
$e^{j2\pi 3/6}$



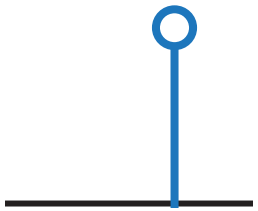
Shift sampling pattern by 1

Combining three ideas

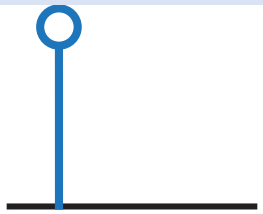
Sparse spectrum



Can recover blue location via phase differences

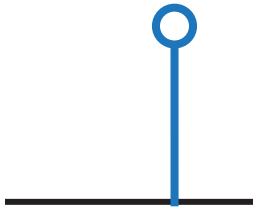


Subsampling by 3 in signal domain



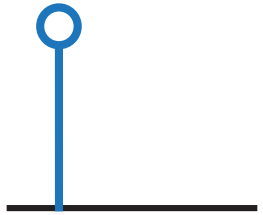
Subsampling by 2 in signal domain

$e^{j2\pi 3/6}$



Shift sampling pattern by 1

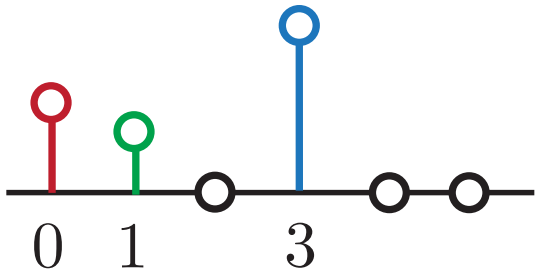
$e^{j2\pi 3/6}$



Shift sampling pattern by 1

Combining three ideas

Sparse spectrum



Recovered all sparse entries

Subsampling by 3 in signal domain

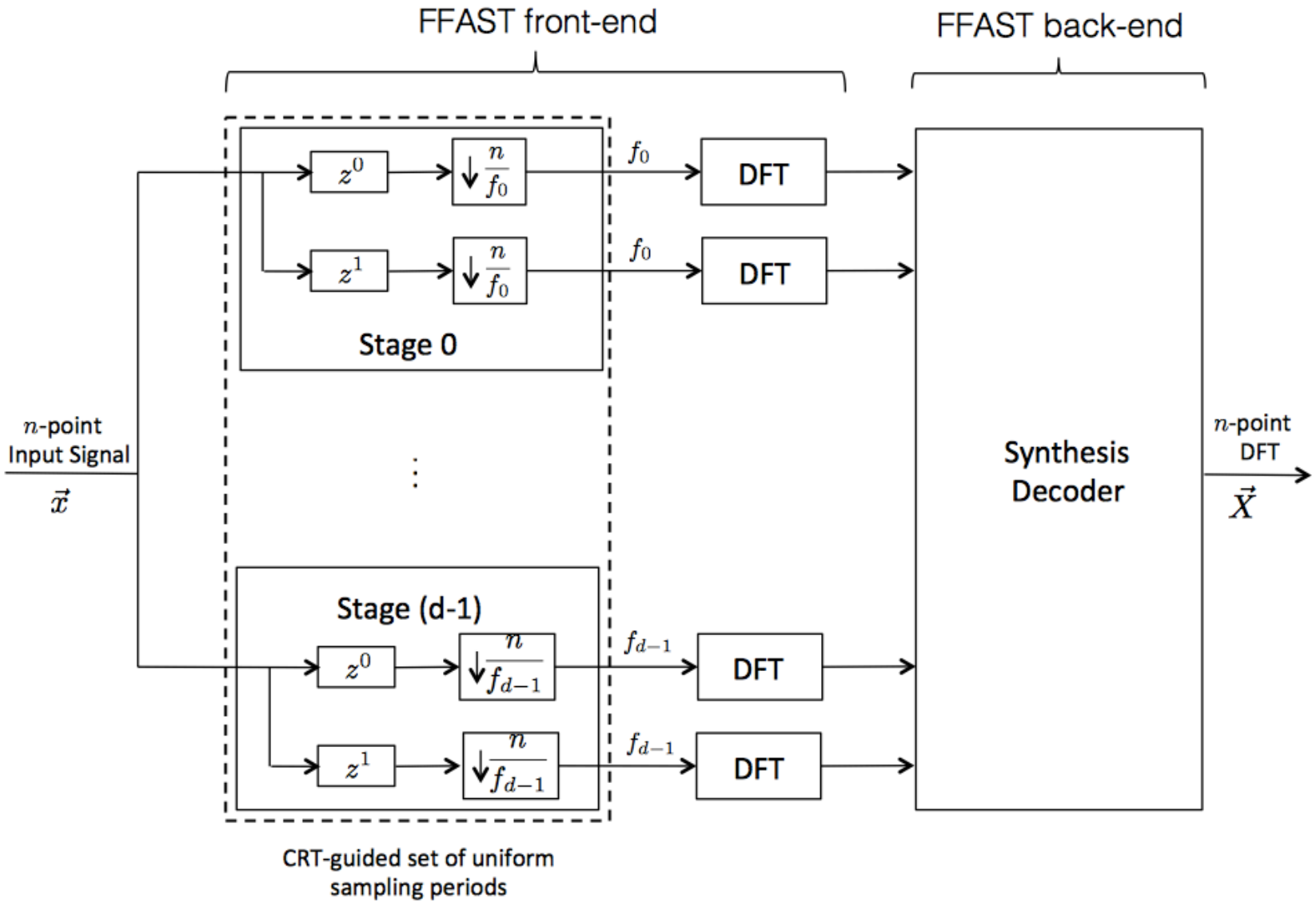
Subsampling by 2 in signal domain

Shift sampling pattern by 1

Shift sampling pattern by 1

FFAST Architecture

Sampling factor: $f_i \sim k$
Number of stages: ~3
Computational complexity: O(k log k)



Sampling Factors

- How to pick sampling factors that gives “different” aliasing patterns?
  - Does undersample by 2 and 4 work?

Sampling Factors

- How to pick sampling factors that gives “different” aliasing patterns?
  - Does undersample by 2 and 4 work?
  - No, undersampling by 4 does not give additional info.
  - Chinese Remainder Theorem:
    - Factors should be relatively co-prime
  - For example, subsample by 5, 6, and 7

Theoretical Guarantee

- Proof techniques from coding theory
- The FFAST algorithm is fast:
  - Uses <4 k samples
  - O(k log k) computation complexity
- Trades off robustness to noise compared to compressed sensing
- For more detail: https://www.eecs.berkeley.edu/~kannanr/project_ffft.html

The Art of Filter Design

Frank Ong

Filter Design

- Given an ideal filter response profile

$$H_{\text{ideal}}(z) = \sum_n h[n] z^{-n}$$

- How can you design $h[n]$ to match $H_{\text{ideal}}(z)$?

Reduce to a linear problem

- Key observation:
  - Filter response is **linear** in h[n]

$$\sum_n h[n] z^{-n}$$

Reduce to a linear problem

- Key observation:
  - Filter response is **linear** in h[n]

$$\sum_n h[n] z^{-n} = \mathbf{z}^T \mathbf{h}$$

$$\mathbf{z} = [1 \; z^{-1} \; z^{-2} \; \ldots \; z^{-n}]^T$$

$$\mathbf{h} = [h[0] \; h[1] \; h[2] \; \ldots \; h[n]]^T$$

Filter design as a linear program

- Discretize z into z_i
- Find $\mathbf{h}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{h} - H_{\text{ideal}}(z_i) \leq \delta$$

Filter design as a linear program

- Discretize z into z_i
- Find $\mathbf{h}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{h} - H_{\text{ideal}}(z_i) \leq \delta$$

- This is a linear program

Magnitude filter design

- Oftentimes, we don’t care about the phase of our filter

Magnitude filter design

- Oftentimes, we don't care about the phase of our filter
- Given an ideal magnitude filter response profile

$$|H_{\text{ideal}}(z)| = |\sum_{n} h[n] z^{-n}|$$

- How can you design $h[n]$ to match $|H_{\text{ideal}}(z)|$?

Magnitude filter design

- Oftentimes, we don't care about the phase of our filter
- Given an ideal magnitude filter response profile

$$|H_{\text{ideal}}(z)| = |\sum_{n} h[n] z^{-n}|$$

- How can you design $h[n]$ to match $|H_{\text{ideal}}(z)|$?

- Is h[n] still linear in magnitude response?

Reduce to a quadratic problem

- Key observation:
  - Magnitude filter response is **quadratic** in h[n]

$$\left| \sum_n h[n] z^{-n} \right|^2$$

Reduce to a quadratic problem

- Key observation:
  - Magnitude filter response is **quadratic** in h[n]

$$\left| \sum_n h[n] z^{-n} \right|^2 = |\mathbf{z}^T \mathbf{h}|^2$$

Reduce to a quadratic problem

- Key observation:
  - Magnitude filter response is **quadratic** in h[n]

$$\left|\sum_n h[n] z^{-n}\right|^2 = |\mathbf{z}^T\mathbf{h}|^2$$

$$= \mathbf{z}^T\mathbf{h}\mathbf{h}^T\mathbf{z}$$

Reduce to a quadratic problem

- Key observation:
  - Magnitude filter response is **quadratic** in h[n]

$$\left|\sum_n h[n]z^{-n}\right|^2 = |\mathbf{z}^T\mathbf{h}|^2$$

$$= \mathbf{z}^T\mathbf{h}\mathbf{h}^T\mathbf{z}$$

$$\mathbf{h}\mathbf{h}^T = \begin{pmatrix} h[0]h[0] & h[0]h[1] & h[0]h[2] \\ h[1]h[0] & h[1]h[1] & h[1]h[2] \\ h[2]h[0] & h[2]h[1] & h[2]h[2] \end{pmatrix}$$

Filter design as a rank-1 quadratic program

- Discretize z into z_i
- Find $\mathbf{h}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{h}\mathbf{h}^\mathbf{T}\mathbf{z_i} - |H_{\text{ideal}}(z_i)|^2 \leq \delta$$

Filter design as a rank-1 quadratic program

- Discretize z into z_i
- Find $\mathbf{h}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{H} \mathbf{z_i} - \left| H_{\text{ideal}}(z_i) \right|^2 \leq \delta$$

$$\mathbf{H} = \mathbf{h}\mathbf{h}^T$$

Filter design as a rank-1 quadratic program

- Discretize z into z_i
- Find $\mathbf{h}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{H} \mathbf{z_i} - |H_{\text{ideal}}(z_i)|^2 \leq \delta$$

$$\mathbf{H} = \mathbf{h}\mathbf{h}^T$$

- The rank-1 constraint is non-convex!

Filter design as a Semi-definite Program

- Discretize z into z_i
- Find $\mathbf{H}$ such that for all z_i you care about

$$-\delta \leq \mathbf{z_i}^T \mathbf{H} \mathbf{z_i} - \left| H_{\text{ideal}}(z_i) \right|^2 \leq \delta$$

$$\mathbf{H} \succeq 0$$

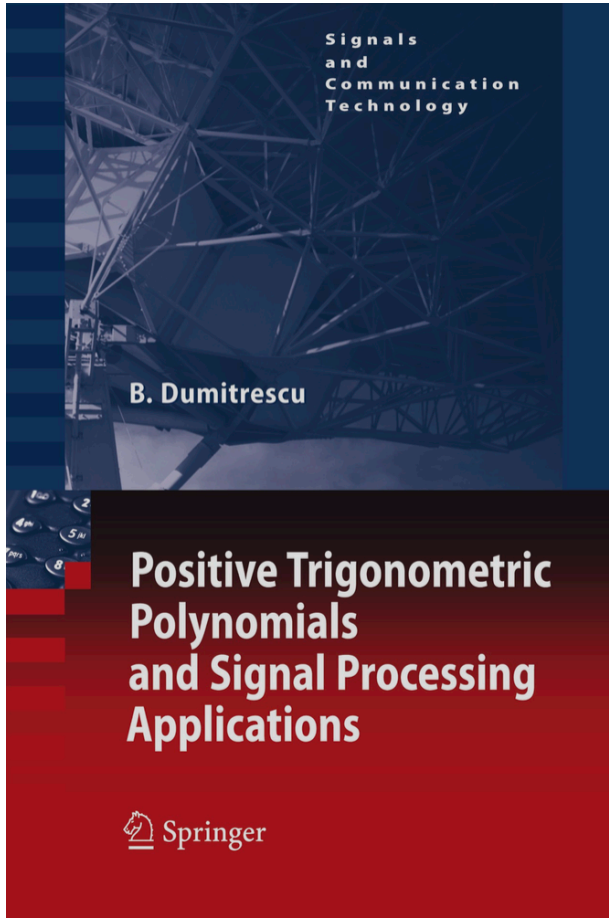
- Relax to a convex semi-definite constraint (lifting)!
- Hot topic in convex optimization

Filter design as a Semi-definite Program

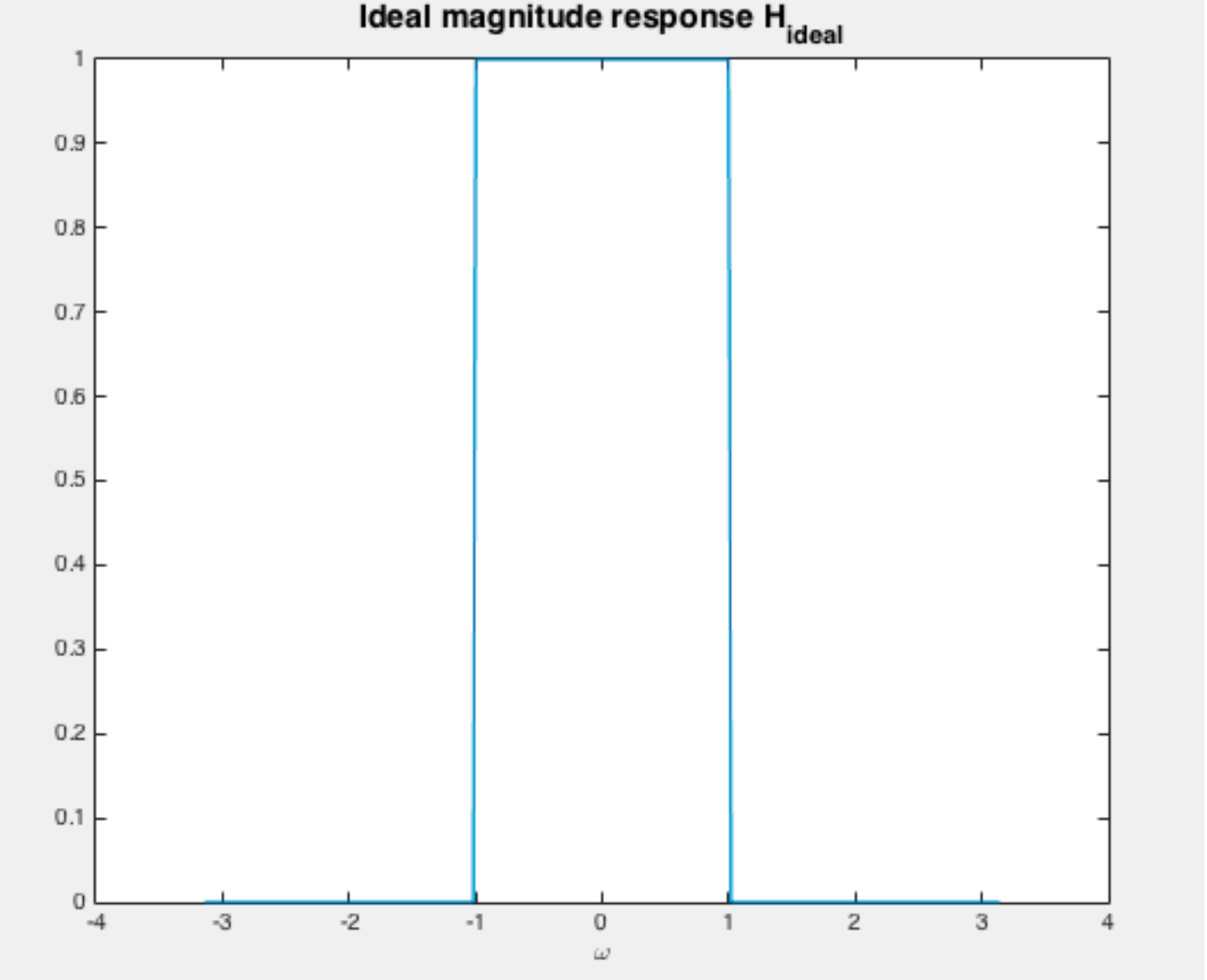
- H is in general not unique

$$\begin{aligned} \underset{H \succeq 0}{\text{maximize}} \quad & H[0][0] \\ \text{subject to} \quad & -\delta \leq \mathbf{z_i}^T \mathbf{H} \mathbf{z_i} - |H_{\text{ideal}}(z_i)|^2 \leq \delta \end{aligned}$$

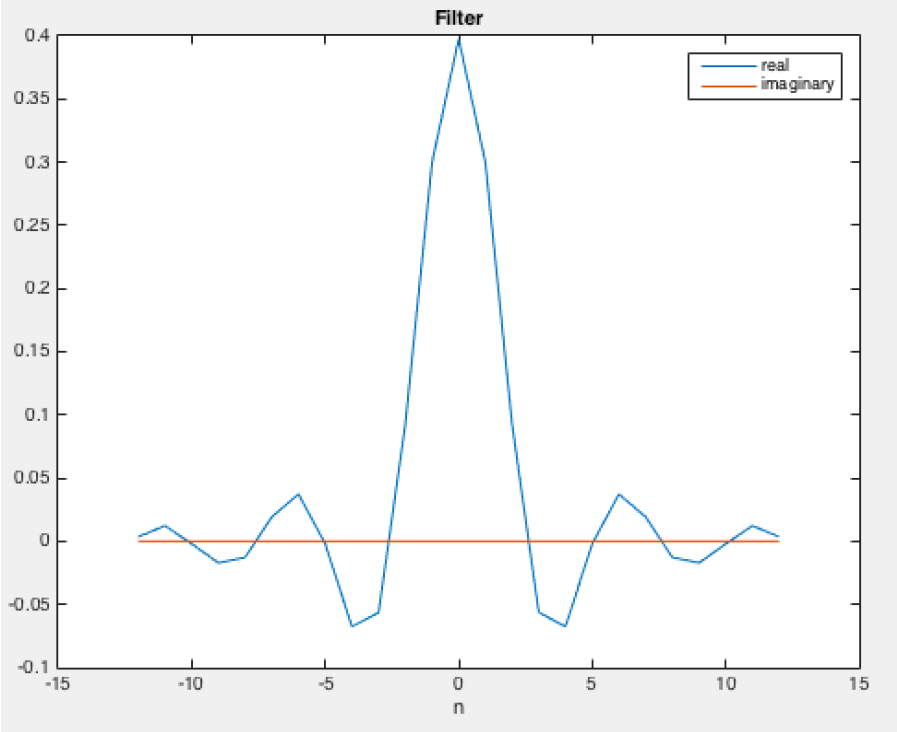
- Outputs the minimum phase $\mathbf{h}$

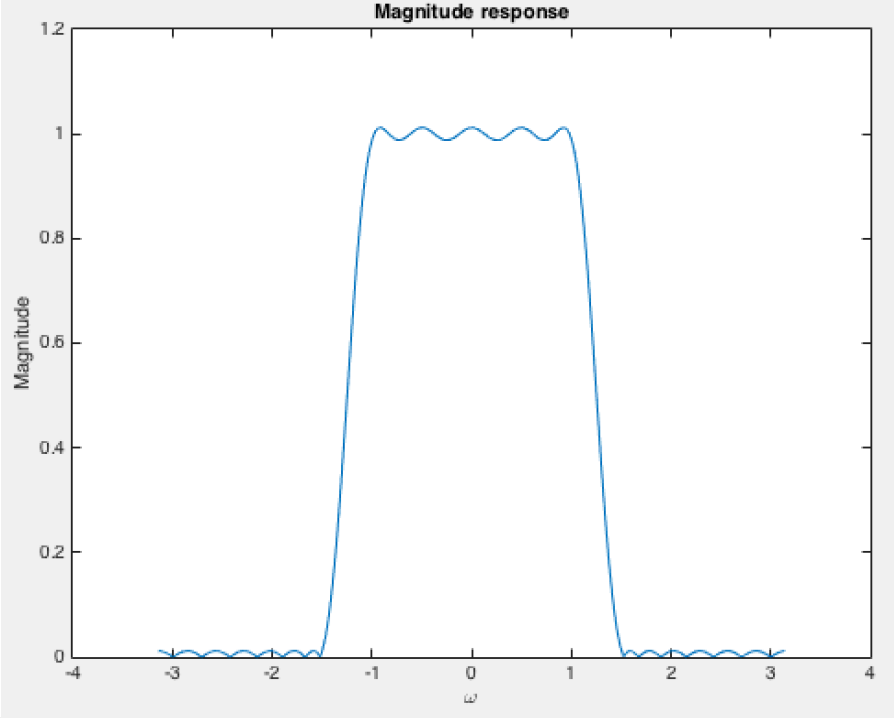


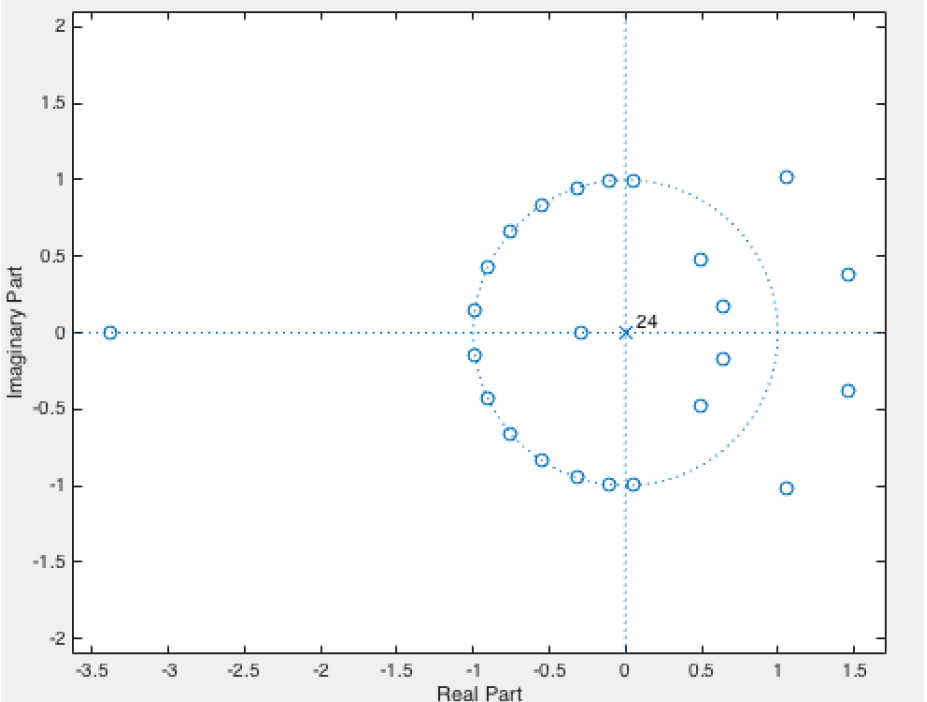
Example


Ideal magnitude response H ideal
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4
ω

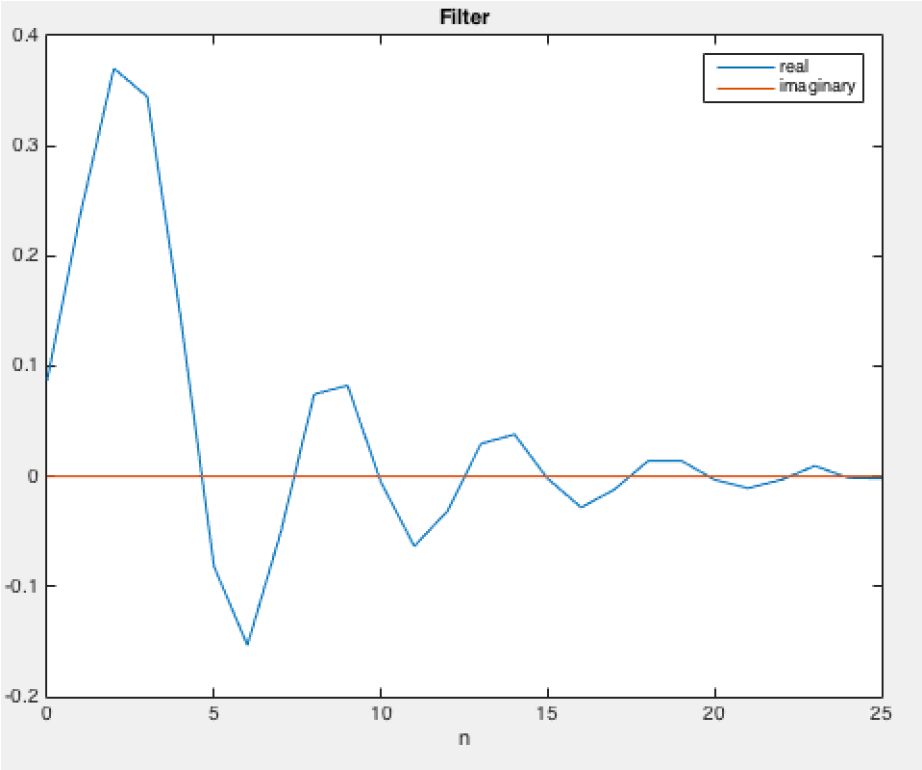
Zero-phase filter using linear program

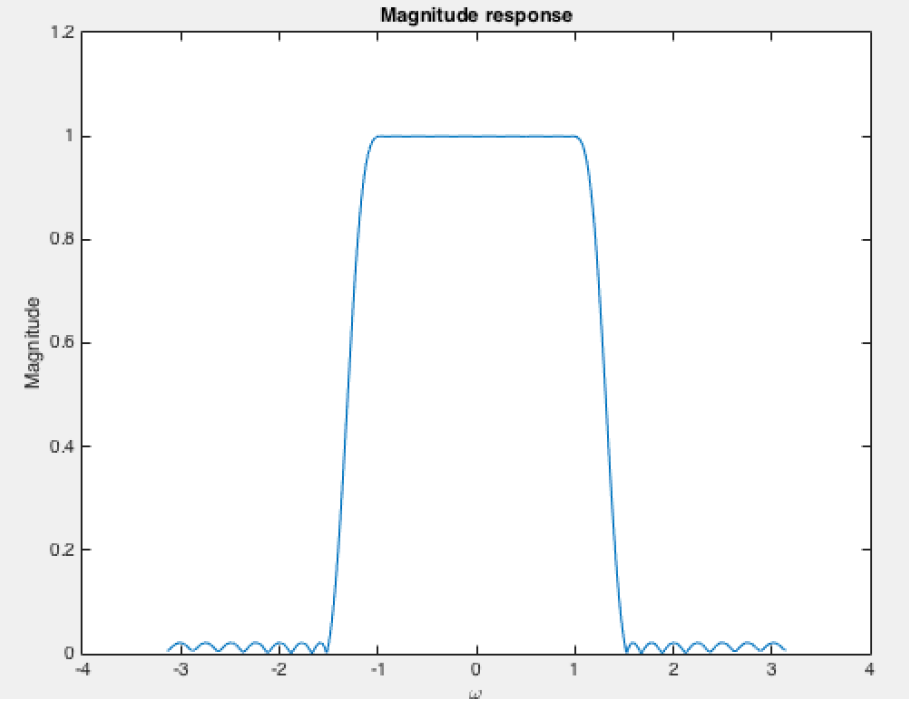


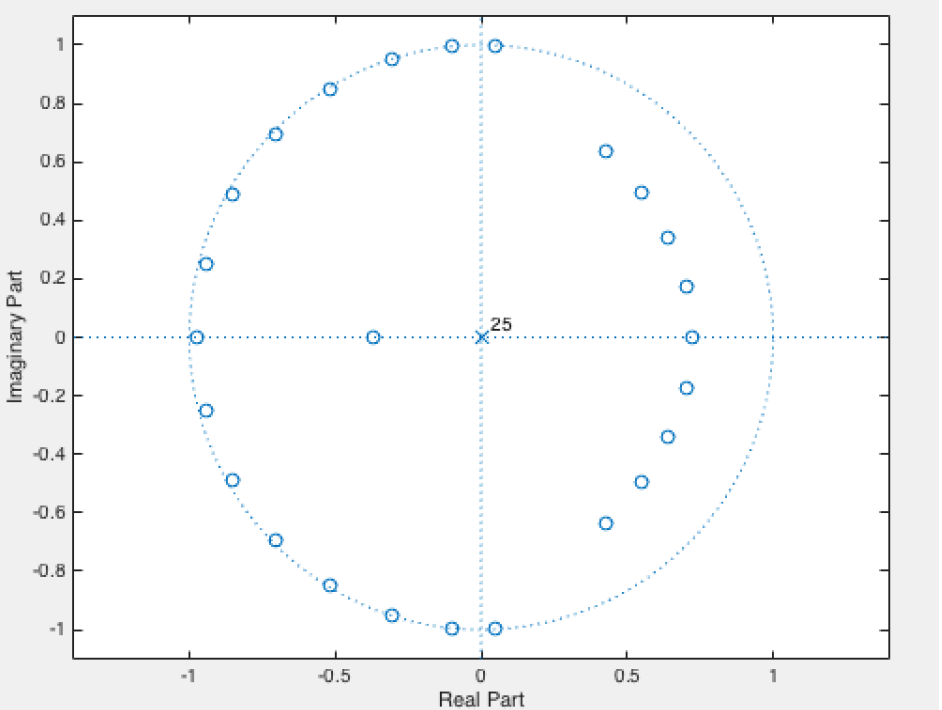




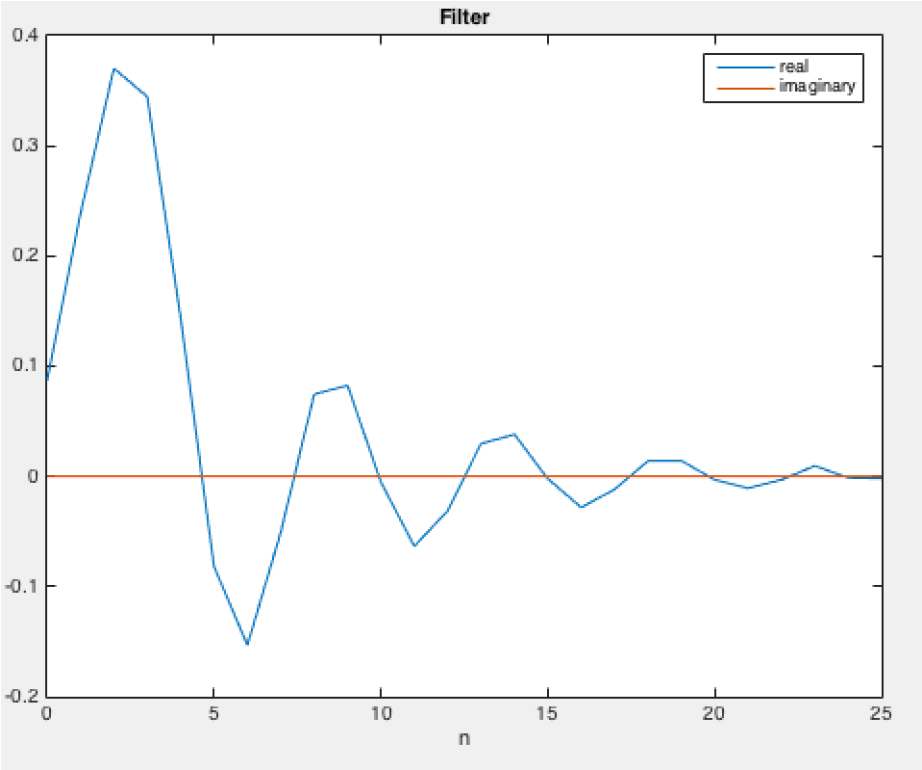
Minimum phase filter using SDP

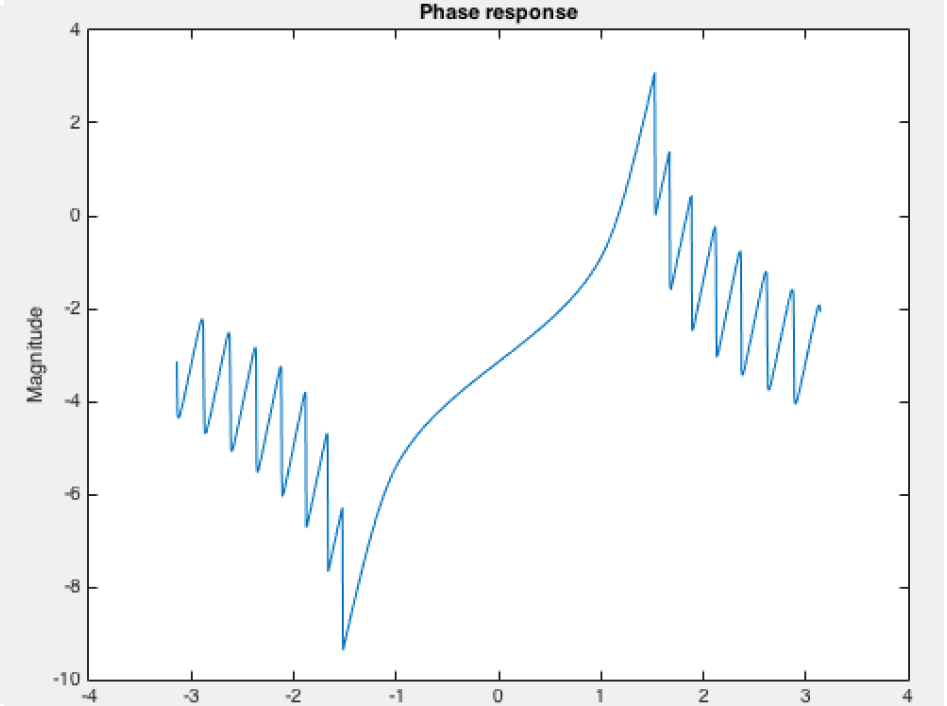


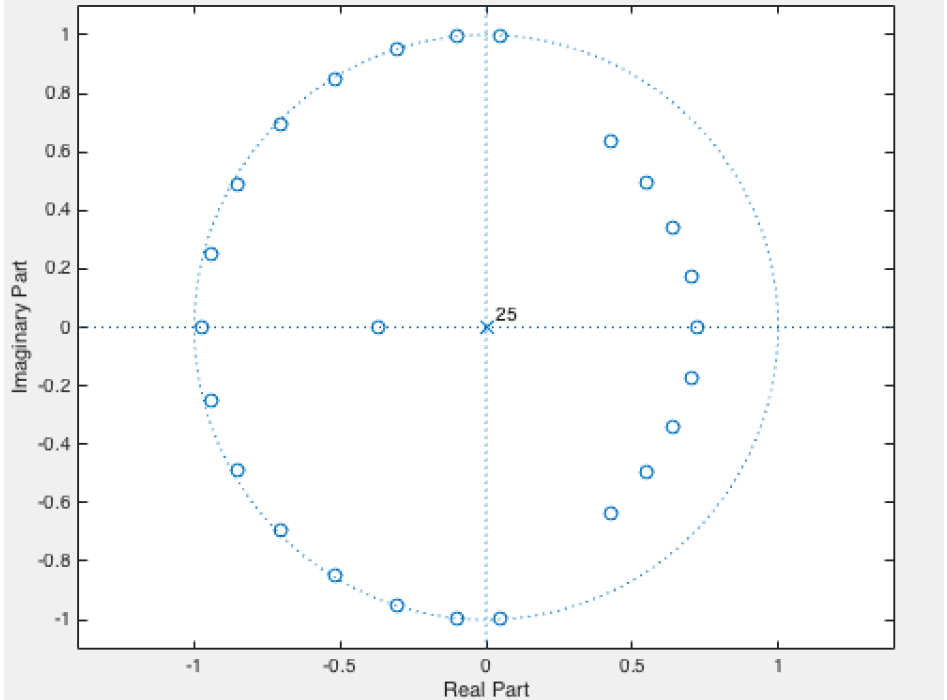




Minimum phase filter using SDP







Github package

- Filter design package using SDP:
- https://github.com/frankong/filter_design
- Uses CVX