Sparse Fourier Transforms

Eric Price

UT Austin

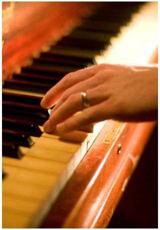
The Fourier Transform

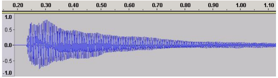
Conversion between time and frequency domains

Time Domain

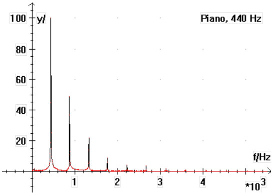
Frequency Domain

Fourier Transform ⟷





Displacement of Air

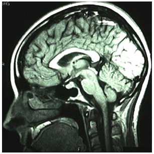


Concert A

The Fourier Transform is Ubiquitous

Audio

Video



Medical Imaging

Radar

GPS

Oil Exploration

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

[T]he method greatly reduces the tediousness of mechanical calculations.

– Carl Friedrich Gauss, 1805

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

> *[T]he method greatly reduces the tediousness of mechanical calculations.*
>
> – Carl Friedrich Gauss, 1805

- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

> *[T]he method greatly reduces the tediousness of mechanical calculations.*
>
> – Carl Friedrich Gauss, 1805

- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]
- Can we do better?

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

> *[T]he method greatly reduces the tediousness of mechanical calculations.*
>
> – Carl Friedrich Gauss, 1805

- By hand: $22 n \log n$ seconds. [Danielson-Lanczos, 1942]
- Can we do *much* better?

Computing the Discrete Fourier Transform

- How to compute $\widehat{x} = Fx$?
- Naive multiplication: $O(n^2)$.
- Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

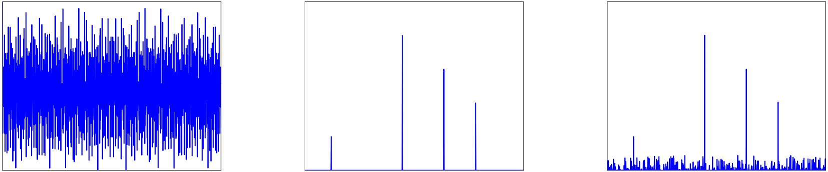
> *[T]he method greatly reduces the tediousness of mechanical calculations.*
>
> – Carl Friedrich Gauss, 1805

- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]
- Can we do *much* better?

When can we compute the Fourier Transform in *sublinear* time?

Idea: Leverage *Sparsity*

Often the Fourier transform is dominated by a small number of peaks:



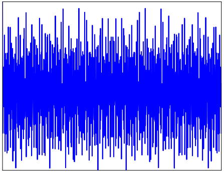
Time Signal

Frequency
(Exactly sparse)

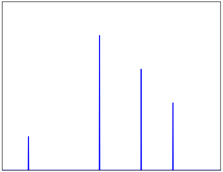
Frequency
(Approximately sparse)

Idea: Leverage *Sparsity*

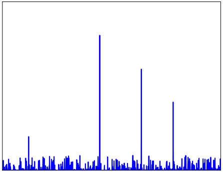
Often the Fourier transform is dominated by a small number of peaks:



Time Signal



Frequency
(Exactly sparse)

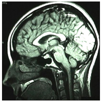


Frequency
(Approximately sparse)

Sparsity is common:

Audio

Video



Medical Imaging

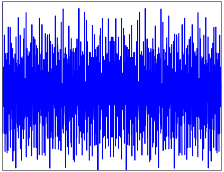
Radar

GPS

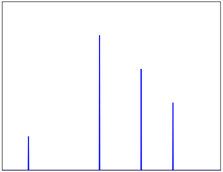
Oil Exploration

Idea: Leverage *Sparsity*

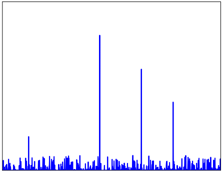
Often the Fourier transform is dominated by a small number of peaks:



Time Signal



Frequency
(Exactly sparse)



Frequency
(Approximately sparse)

Sparsity is common:

Goal of this talk: *sparse* Fourier transforms

Faster Fourier Transform on sparse data.

Recent Theory and Applied Work

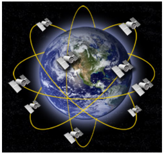
- Sparse Fourier Transform in the Discrete Setting
  - Gilbert-Guha-Indyk-Muthukrishnan-Strauss, 02
  - Gilbert-Muthukrishnan-Strauss, 05
  - Hassanieh-Indyk-Katabi-Price, 12
  - Indyk-Kapralov, 14

Recent Theory and Applied Work

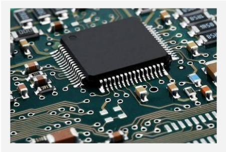
- Sparse Fourier Transform in the Discrete Setting
  - Gilbert-Guha-Indyk-Muthukrishnan-Strauss, 02
  - Gilbert-Muthukrishnan-Strauss, 05
  - Hassanieh-Indyk-Katabi-Price, 12
  - Indyk-Kapralov, 14
- Sparse Fourier Transform in the Continuous Setting
  - Boufounos-Cevher-Gilbert-Li-Strauss, 12
  - Price-Song, 15

Recent Theory and Applied Work

- Sparse Fourier Transform in the Discrete Setting
  - Gilbert-Guha-Indyk-Muthukrishnan-Strauss, 02
  - Gilbert-Muthukrishnan-Strauss, 05
  - Hassanieh-Indyk-Katabi-Price, 12
  - Indyk-Kapralov, 14
- Sparse Fourier Transform in the Continuous Setting
  - Boufounos-Cevher-Gilbert-Li-Strauss, 12
  - Price-Song, 15
- Applications



Faster GPS ... Fourier ...
Hassanieh et al.
MOBICOM'12



... Fourier ... Chip ...
Abari et al.
ISSCC'12

... Chemical ... Imaging ...
Andronesi et al.
ENC'14

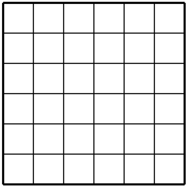
Light ... Continuous Fourier...
Shi et al.
SIGGRAPH'15

Kinds of discrete Fourier transform

- 1d Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

Kinds of discrete Fourier transform



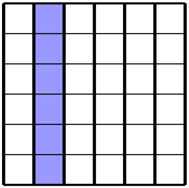
- $1d$ Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

- $2d$ Fourier Transform: $x \in \mathbb{C}^{n_1 \times n_2}$, $\omega_i = e^{2\pi \mathbf{i}/n_i}$, want

$$\widehat{x}_{i_1,i_2} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} x_{j_1,j_2}$$

Kinds of discrete Fourier transform



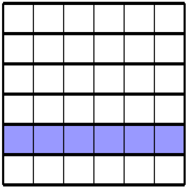
- $1d$ Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

- $2d$ Fourier Transform: $x \in \mathbb{C}^{n_1 \times n_2}$, $\omega_i = e^{2\pi \mathbf{i}/n_i}$, want

$$\widehat{x}_{i_1,i_2} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} x_{j_1,j_2}$$

Kinds of discrete Fourier transform



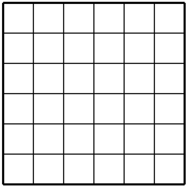
- 1*d* Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

- 2*d* Fourier Transform: $x \in \mathbb{C}^{n_1 \times n_2}$, $\omega_i = e^{2\pi \mathbf{i}/n_i}$, want

$$\widehat{x}_{i_1,i_2} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} x_{j_1,j_2}$$

Kinds of discrete Fourier transform



- $1d$ Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

- $2d$ Fourier Transform: $x \in \mathbb{C}^{n_1 \times n_2}$, $\omega_i = e^{2\pi \mathbf{i}/n_i}$, want

$$\widehat{x}_{i_1,i_2} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} x_{j_1,j_2}$$

  - If n_1, n_2 are relatively prime, *equivalent* to $1d$ transform of $\mathbb{C}^{n_1 n_2}$

Kinds of discrete Fourier transform

- $1d$ Fourier transform: $x \in \mathbb{C}^n$, $\omega = e^{2\pi \mathbf{i}/n}$, want

$$\widehat{x}_i = \sum_{j=1}^{n} \omega^{ij} x_j$$

- $2d$ Fourier Transform: $x \in \mathbb{C}^{n_1 \times n_2}$, $\omega_i = e^{2\pi \mathbf{i}/n_i}$, want

$$\widehat{x}_{i_1,i_2} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \omega_1^{i_1 j_1} \omega_2^{i_2 j_2} x_{j_1,j_2}$$

  - If n_1, n_2 are relatively prime, *equivalent* to $1d$ transform of $\mathbb{C}^{n_1 n_2}$
- Hadamard transform: $x \in \mathbb{C}^{2 \times 2 \times \cdots \times 2}$:

$$\widehat{x}_i = \sum_{j}^{n} (-1)^{\langle i,j \rangle} x_j$$

Generic Algorithm Outline

- Goal: given access to x, compute $\overline{x} \approx \widehat{x}$
  - Exact case: $\widehat{x}$ is k-sparse, $\overline{x} = \widehat{x}$ (maybe to $\log n$ bits of precision)

Generic Algorithm Outline

- Goal: given access to x, compute $\overline{x} \approx \widehat{x}$
  - Exact case: $\widehat{x}$ is k-sparse, $\overline{x} = \widehat{x}$ (maybe to $\log n$ bits of precision)
  - Approximate case:

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

Generic Algorithm Outline

- Goal: given access to x, compute $\overline{x} \approx \widehat{x}$
  - Exact case: $\widehat{x}$ is k-sparse, $\overline{x} = \widehat{x}$ (maybe to $\log n$ bits of precision)
  - Approximate case:

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

  - With "good" probability.

Generic Algorithm Outline

- Goal: given access to x, compute $\overline{x} \approx \widehat{x}$
  - Exact case: $\widehat{x}$ is k-sparse, $\overline{x} = \widehat{x}$ (maybe to $\log n$ bits of precision)
  - Approximate case:

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1 + \epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

  - With "good" probability.

1. Algorithm for $k = 1$ (exact or approximate)

Generic Algorithm Outline

- Goal: given access to x, compute $\overline{x} \approx \widehat{x}$
  - Exact case: $\widehat{x}$ is k-sparse, $\overline{x} = \widehat{x}$ (maybe to $\log n$ bits of precision)
  - Approximate case:

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

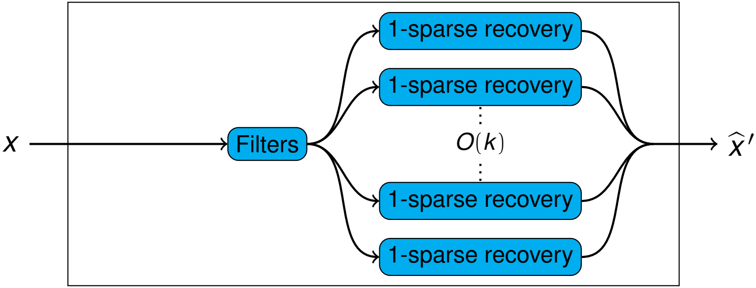
  - With "good" probability.

1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case

Generic Algorithm Outline

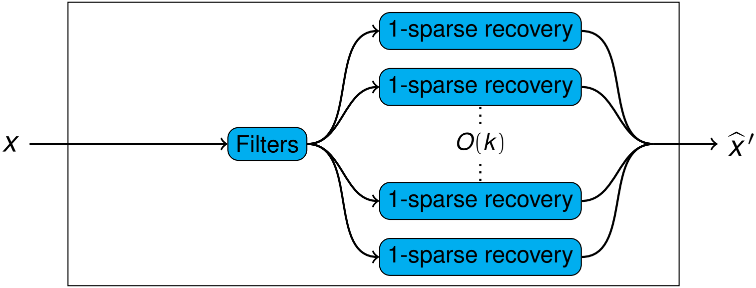
1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case

Generic Algorithm Outline



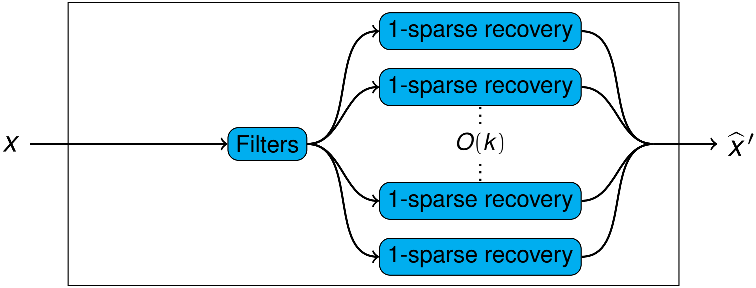
1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case

Generic Algorithm Outline



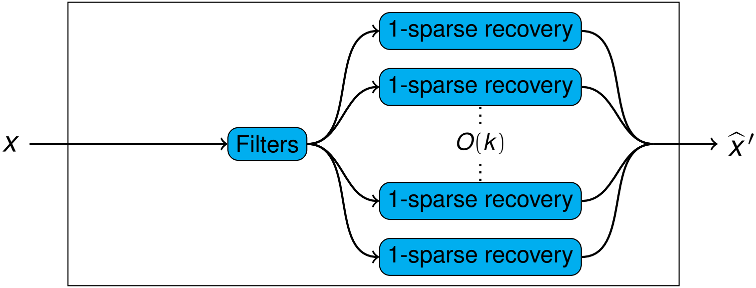
1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case
   - Split $\widehat{x}$ into $O(k)$ “random” parts

Generic Algorithm Outline



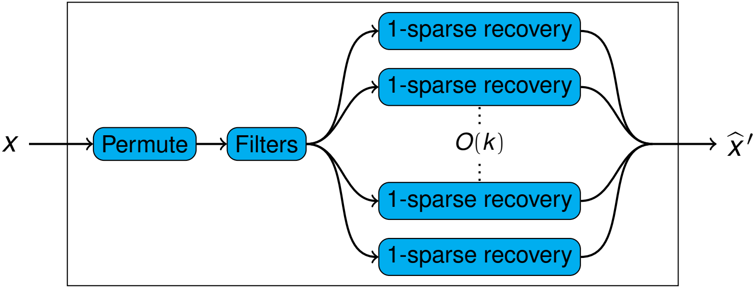
1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case
   - Split $\widehat{x}$ into $O(k)$ "random" parts
   - Can sample time domain of the parts.

Generic Algorithm Outline



1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case
   - Split $\widehat{x}$ into $O(k)$ "random" parts
   - Can sample time domain of the parts.
     - $O(k \log k)$ time to get one sample from each of the k parts.

Generic Algorithm Outline



1. Algorithm for $k = 1$ (exact or approximate)
2. Method to reduce to $k = 1$ case
   - Split $\widehat{x}$ into $O(k)$ "random" parts
   - Can sample time domain of the parts.
     - $O(k \log k)$ time to get one sample from each of the k parts.
3. Finds "most" of signal; repeat on residual

Talk Outline

1. Algorithm for $k = 1$

Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1

Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1
3. Putting it together

Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1
3. Putting it together
4. Continuous setting

Talk Outline

1. Algorithm for $k = 1$

2. Reducing k to 1

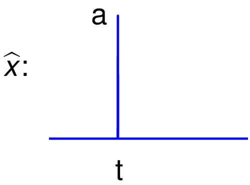
3. Putting it together

4. Continuous setting

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.

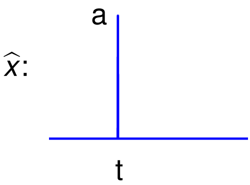


$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.



$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

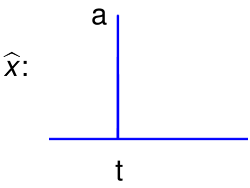
- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.

$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$



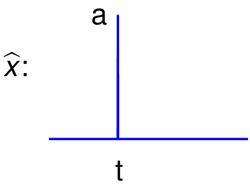
- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

$$x_0 = a$$

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.



$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

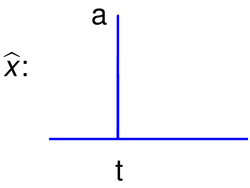
$$x_0 = a \qquad\qquad x_1 = a\omega^t$$

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.

$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$



- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

$$x_0 = a \qquad\qquad x_1 = a\omega^t$$

- $x_1/x_0 = \omega^t \implies t$.

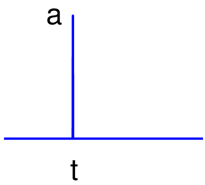
Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.

$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

$\widehat{x}$:



- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

$$x_0 = a \qquad\qquad x_1 = a\omega^t$$

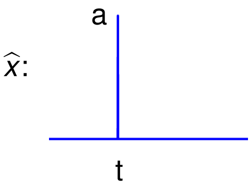
- $x_1/x_0 = \omega^t \implies t$. ■

Algorithm for $k = 1$: one dimension, exact case

Lemma

We can compute a 1-sparse $\widehat{x}$ in $O(1)$ time.

$$\widehat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$



- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

$$x_0 = a \qquad\qquad x_1 = a\omega^t$$

- $x_1/x_0 = \omega^t \implies t$. ∎
- (Related to OFDM, Prony's method, matrix pencil.)

Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t| / \|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.

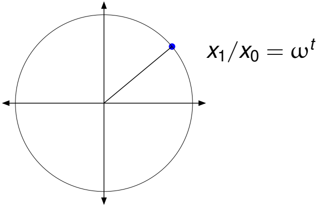
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t|/\|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



- With exact sparsity: $\log n$ bits in a single measurement.

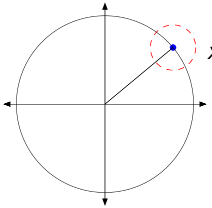
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t|/\|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.

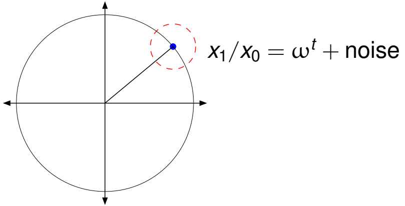
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t| / \|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose $\Theta(\log n)$ time shifts c to recover i.

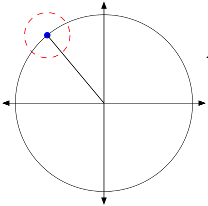
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t| / \|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



$$x_{c_2}/x_0 = \omega^{c_2 t} + \text{noise}$$

- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose $\Theta(\log n)$ time shifts c to recover i.

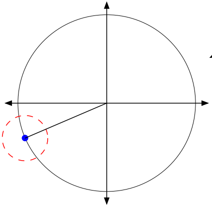
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t|/\|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



$x_{c_3}/x_0 = \omega^{c_3 t} + \text{noise}$

- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose $\Theta(\log n)$ time shifts c to recover i.

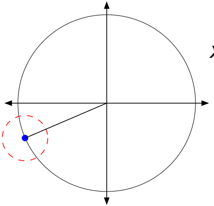
Algorithm for $k = 1$: one dimension, approximate case

Lemma

Suppose $\widehat{x}$ is approximately 1-sparse:

$$|\widehat{x}_t| / \|\widehat{x}\|_2 \geqslant 90\%.$$

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.



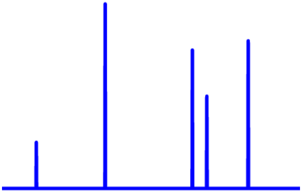
- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose $\Theta(\log n)$ time shifts c to recover i.
- Error correcting code with efficient recovery $\implies$ lemma. ∎

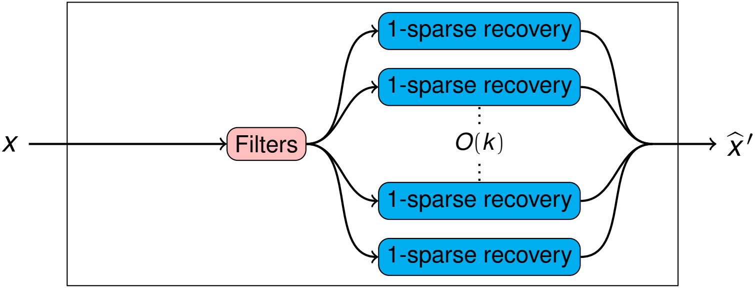
Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1
3. Putting it together
4. Continuous setting

Algorithm for general k

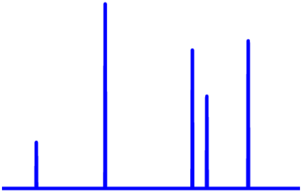
- Reduce general k to $k = 1$.

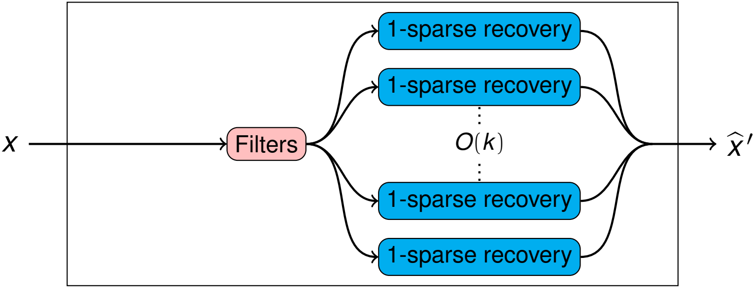




Algorithm for general k

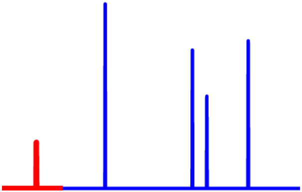
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.

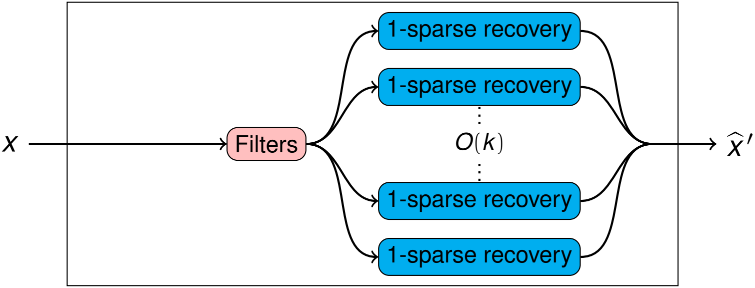




Algorithm for general k

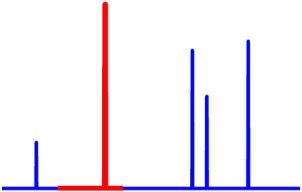
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.

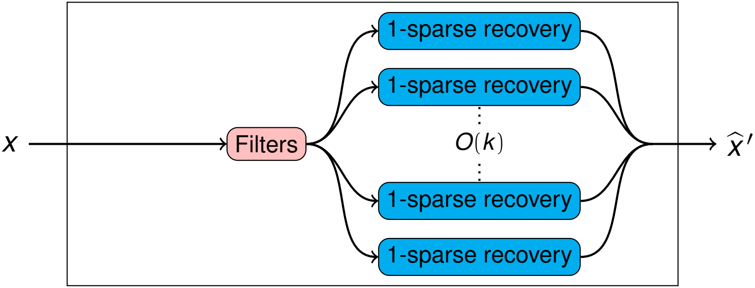




Algorithm for general k

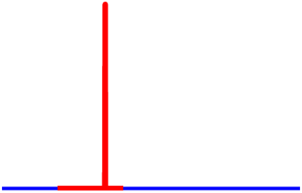
- Reduce general k to $k = 1$.
- “Filters”: partition frequencies into $O(k)$ buckets.

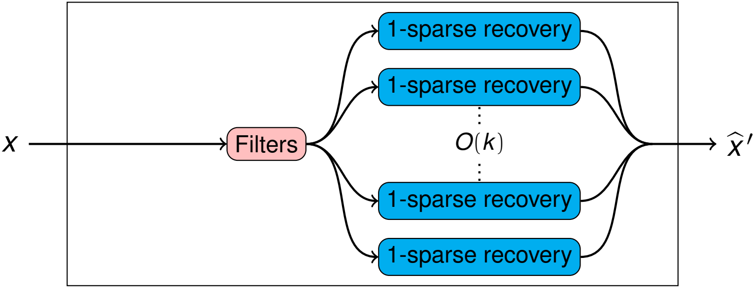




Algorithm for general k

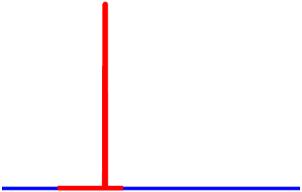
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.

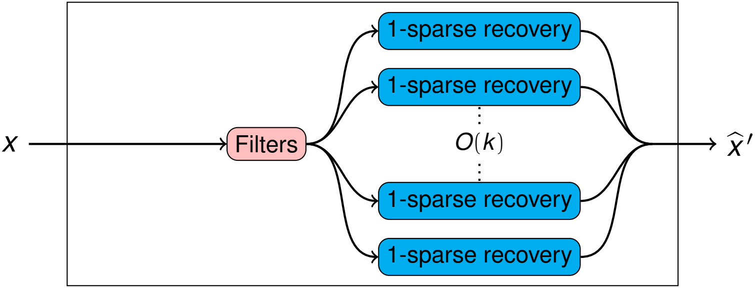




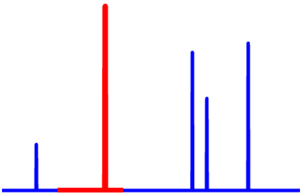
Algorithm for general k

- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm

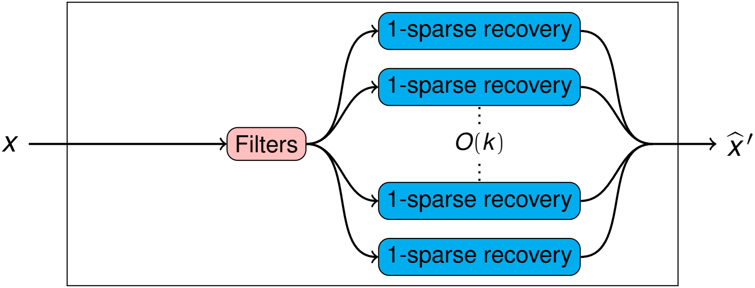




Algorithm for general k

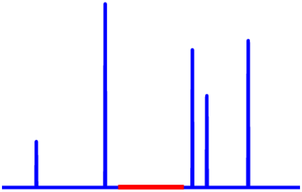


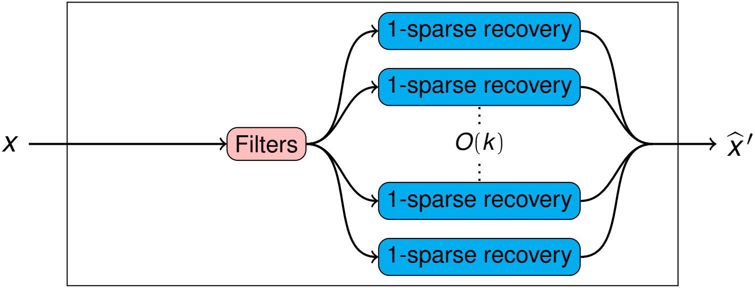
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.



Algorithm for general k

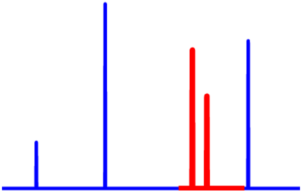
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.

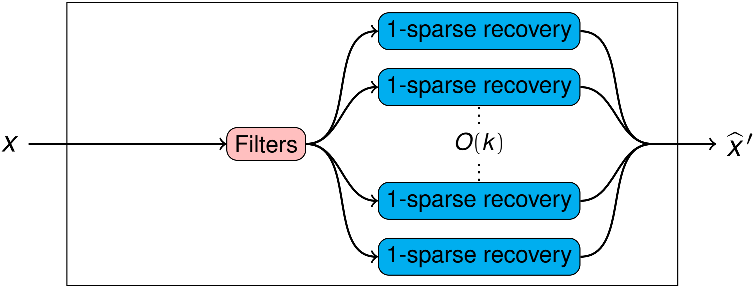




Algorithm for general k

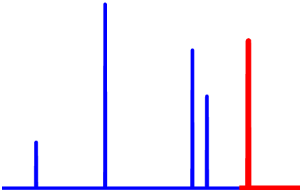
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.

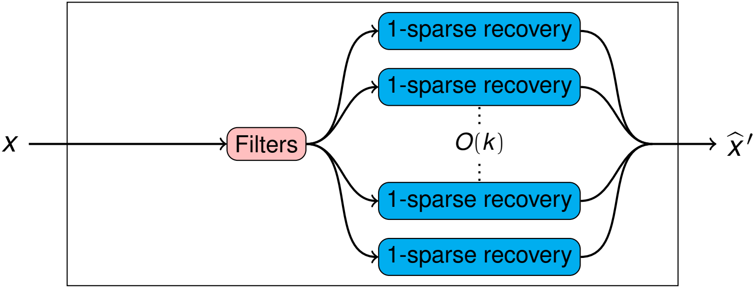




Algorithm for general k

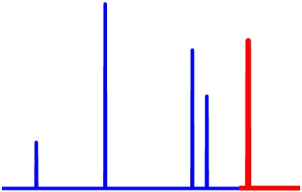
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.

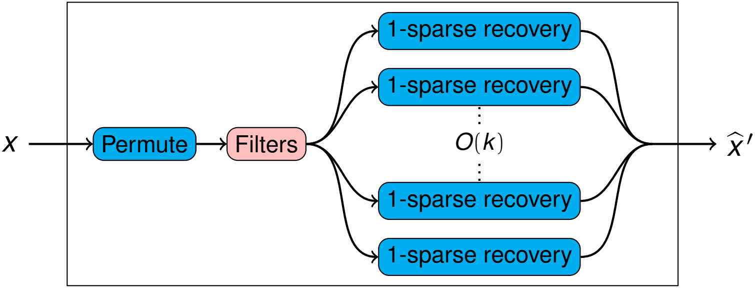




Algorithm for general k

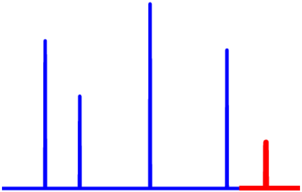
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.
- Random permutation

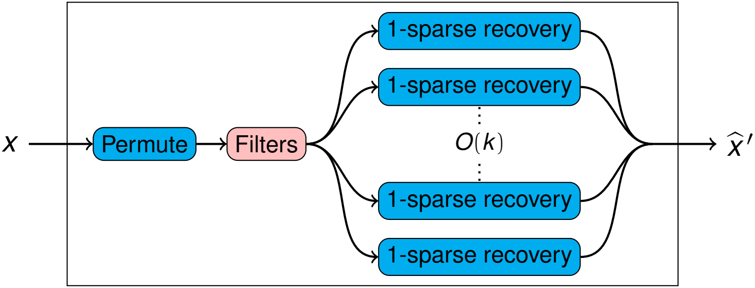




Algorithm for general k

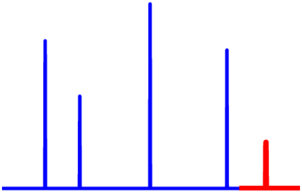
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.
- Random permutation

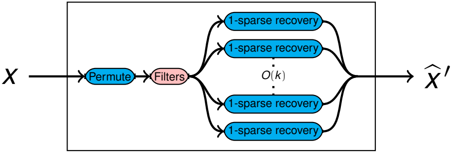




Algorithm for general k

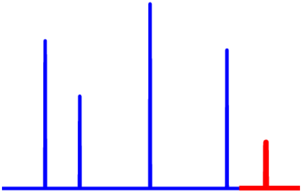
- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.
- Random permutation

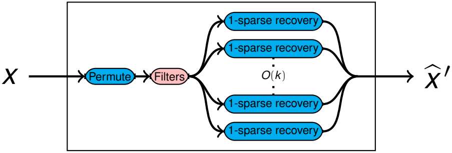




Algorithm for general k

- Reduce general k to $k = 1$.
- "Filters": partition frequencies into $O(k)$ buckets.
  - Sample from time domain of each bucket with $O(\log n)$ overhead.
  - Recovered by $k = 1$ algorithm
- Most frequencies alone in bucket.
- Random permutation



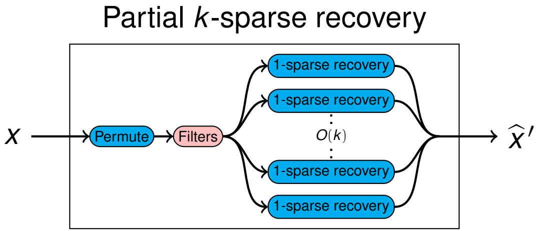


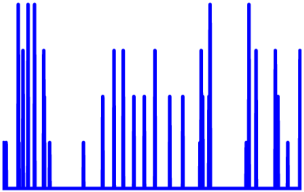
Recovers *most* of $\widehat{x}$:

Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Going from finding most coordinates to finding all $\widehat{x}$



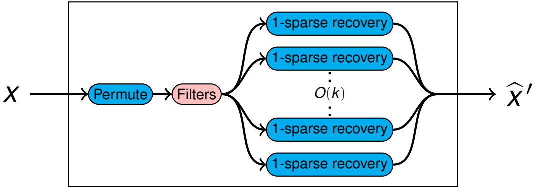


Lemma (Partial sparse recovery)

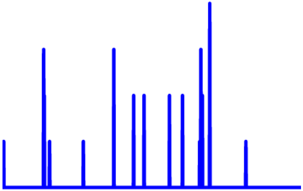
In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Going from finding most coordinates to finding all

Partial k-sparse recovery



$\widehat{x} - \widehat{x}'$

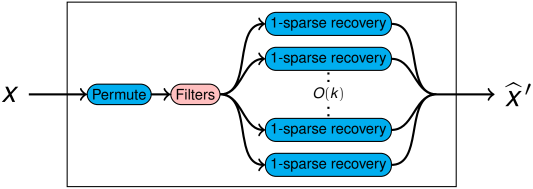


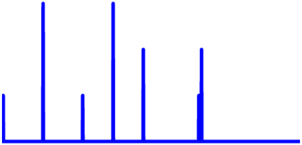
Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Going from finding most coordinates to finding all

Partial k-sparse recovery





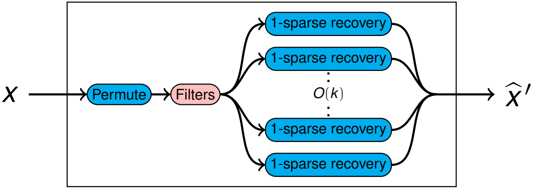
Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Repeat, $k \to k/2 \to k/4 \to \cdots$

Going from finding most coordinates to finding all

Partial k-sparse recovery



Lemma (Partial sparse recovery)

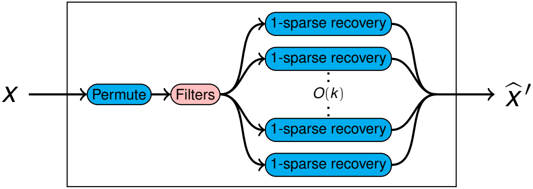
In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

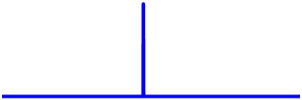
Repeat, $k \to k/2 \to k/4 \to \cdots$

Going from finding most coordinates to finding all

$\widehat{x} - \widehat{x}'$

Partial k-sparse recovery





Lemma (Partial sparse recovery)

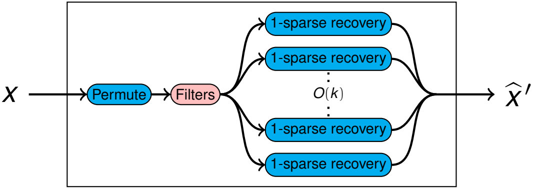
In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Repeat, $k \to k/2 \to k/4 \to \cdots$

Going from finding most coordinates to finding all

$\widehat{x}-\widehat{x}'$

Partial k-sparse recovery



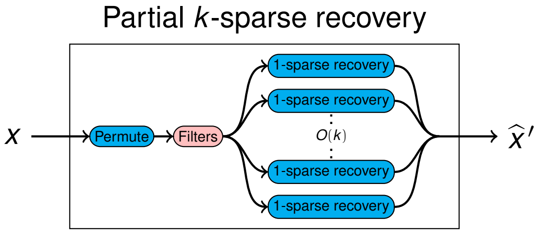
Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x}-\widehat{x}'$ is $k/2$-sparse.

Repeat, $k \to k/2 \to k/4 \to \cdots$

Going from finding most coordinates to finding all

$\widehat{x} - \widehat{x}'$



Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

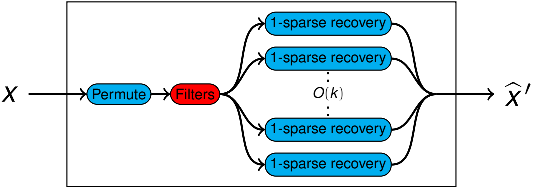
Repeat, $k \to k/2 \to k/4 \to \cdots$

Theorem

We can compute $\widehat{x}$ in $O(k \log n)$ expected time.

Going from finding most coordinates to finding all $\widehat{x} - \widehat{x}'$

Partial k-sparse recovery



Lemma (Partial sparse recovery)

In $O(k \log n)$ expected time, we can compute an estimate $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.

Repeat, $k \to k/2 \to k/4 \to \cdots$

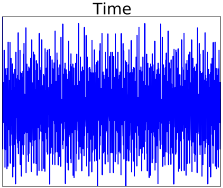
Theorem

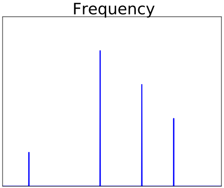
We can compute $\widehat{x}$ in $O(k \log n)$ expected time.

Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1
3. Putting it together
4. Continuous setting

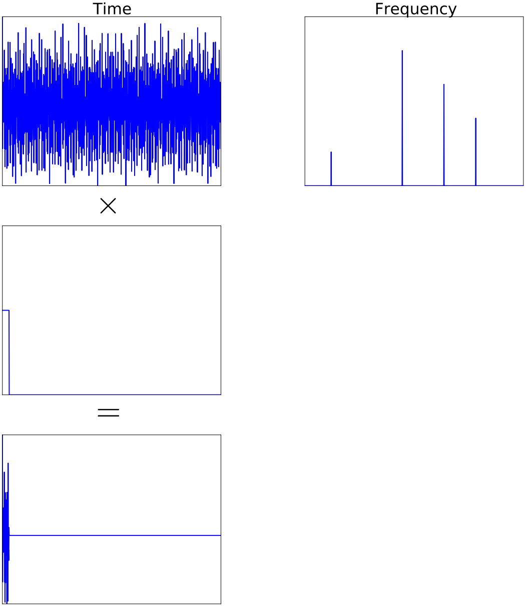
How can you hope for sublinear time?





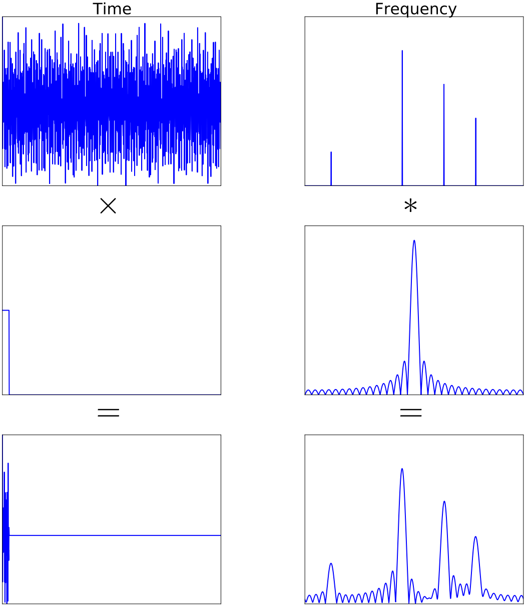
n-dimensional DFT:
$O(n \log n)$
$x \to \widehat{x}$

How can you hope for sublinear time?



n-dimensional DFT:
$O(n \log n)$
$x \rightarrow \widehat{x}$

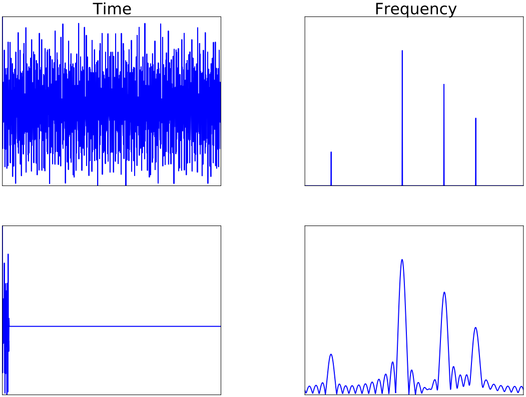
How can you hope for sublinear time?



n-dimensional DFT:
$O(n \log n)$
$x \to \widehat{x}$

n-dimensional DFT of first
k terms: $O(n \log n)$
$x \cdot \text{rect} \to \widehat{x} * \text{sinc}$.

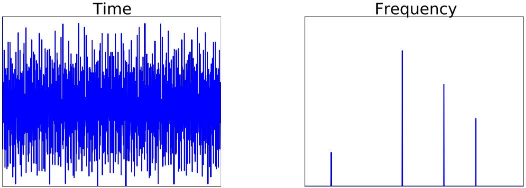
How can you hope for sublinear time?



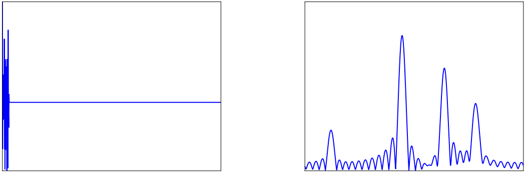
n-dimensional DFT:
$O(n \log n)$
$x \rightarrow \widehat{x}$

n-dimensional DFT of first k terms: $O(n \log n)$
$x \cdot \text{rect} \rightarrow \widehat{x} * \text{sinc}$.

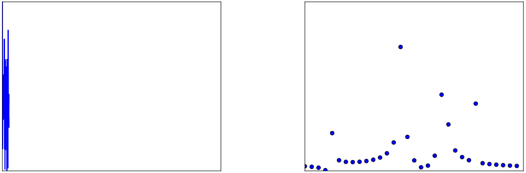
How can you hope for sublinear time?



n-dimensional DFT:
$O(n \log n)$
$x \to \widehat{x}$

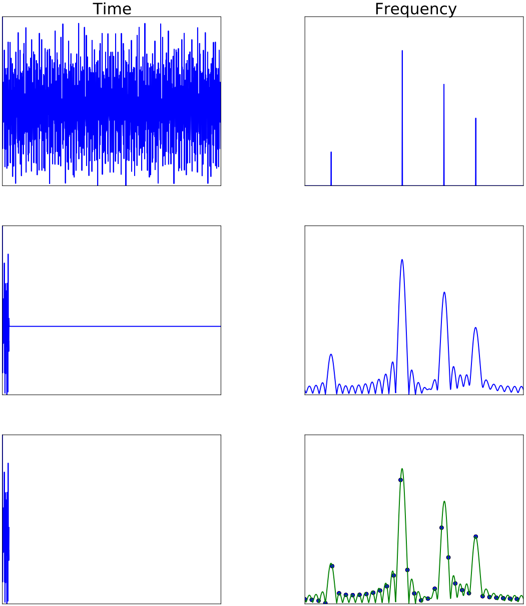


n-dimensional DFT of first k terms: $O(n \log n)$
$x \cdot \text{rect} \to \widehat{x} * \text{sinc}$.



k-dimensional DFT of first k terms: $O(B \log B)$
$\text{alias}(x \cdot \text{rect}) \to \text{subsample}(\widehat{x} * \text{sinc})$.

How can you hope for sublinear time?



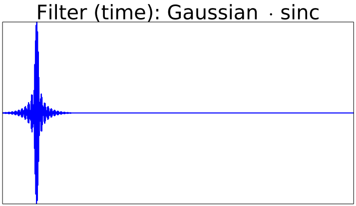
n-dimensional DFT: $O(n \log n)$
$x \to \widehat{x}$

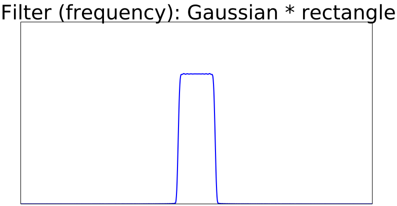
n-dimensional DFT of first k terms: $O(n \log n)$
$x \cdot \text{rect} \to \widehat{x} * \text{sinc}$.

k-dimensional DFT of first k terms: $O(B \log B)$
$\text{alias}(x \cdot \text{rect}) \to \text{subsample}(\widehat{x} * \text{sinc})$.

Use a better filter

GMS05, HIKP12, IKP14, IK14

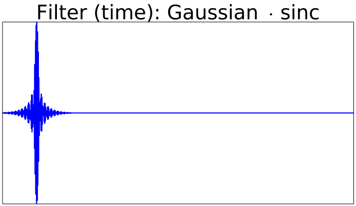


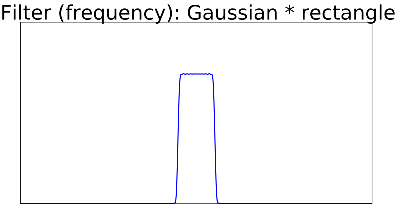


- Filter: sparse F for which $\widehat{F}$ is large on an *interval*.

Use a better filter

GMS05, HIKP12, IKP14, IK14





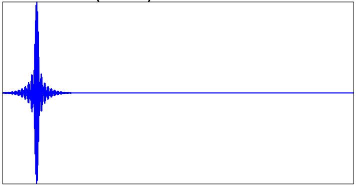
- Filter: sparse F for which $\widehat{F}$ is large on an *interval*.
- We can permute the frequencies:

$$x'_i = x_{\sigma i} \implies \widehat{x}_i = \widehat{x}_{\sigma^{-1} i}$$

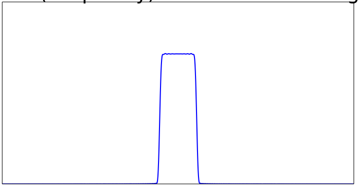
Use a better filter

GMS05, HIKP12, IKP14, IK14

Filter (time): Gaussian $\cdot$ sinc



Filter (frequency): Gaussian * rectangle

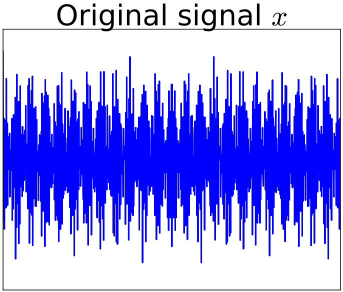


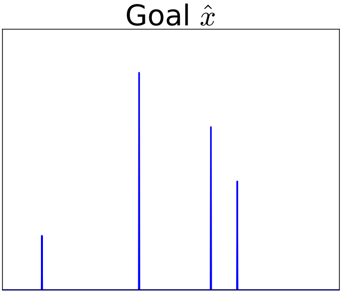
- Filter: sparse F for which $\widehat{F}$ is large on an *interval*.
- We can permute the frequencies:

$$x'_i = x_{\sigma i} \implies \widehat{x'}_i = \widehat{x}_{\sigma^{-1} i}$$

- Allows us to convert worst case to random case.

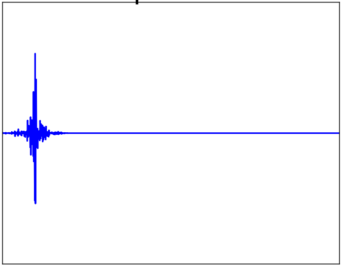
Algorithm for *exactly sparse* signals



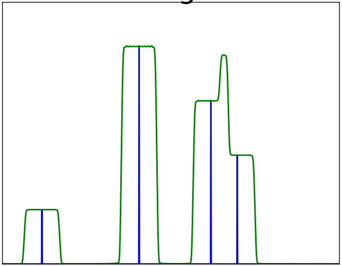


Algorithm for *exactly sparse* signals

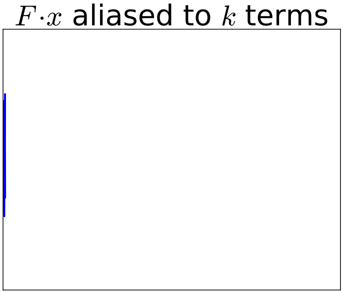
Computed $F \cdot x$

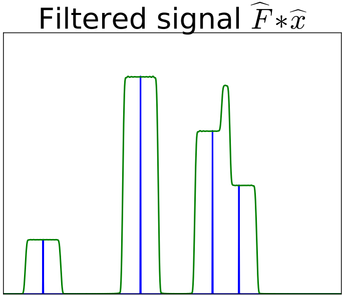


Filtered signal $\widehat{F} * \widehat{x}$



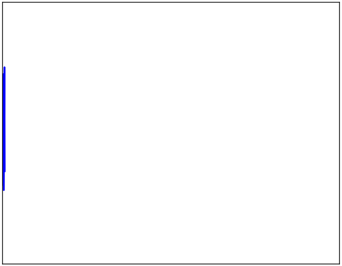
Algorithm for *exactly sparse* signals



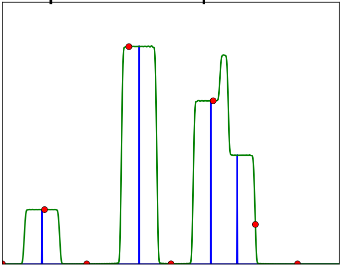


Algorithm for *exactly sparse* signals

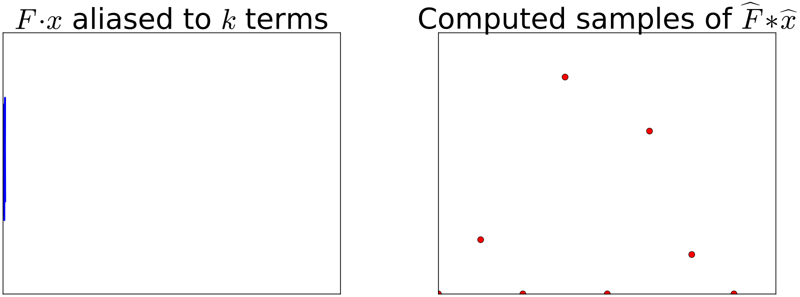
$F \cdot x$ aliased to k terms



Computed samples of $\widehat{F} * \widehat{x}$

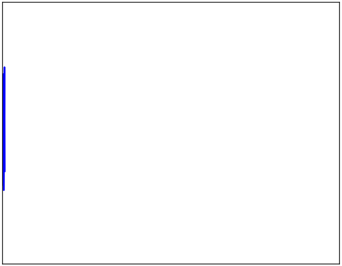


Algorithm for *exactly sparse* signals

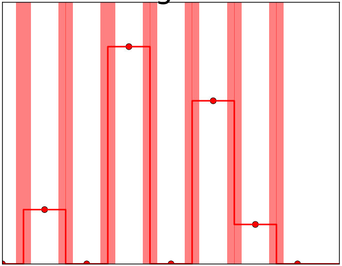


Algorithm for *exactly sparse* signals

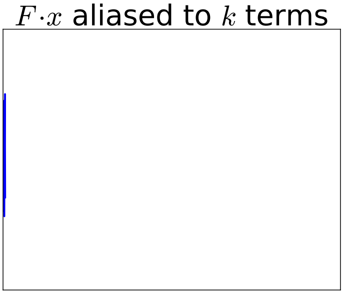
$F \cdot x$ aliased to k terms

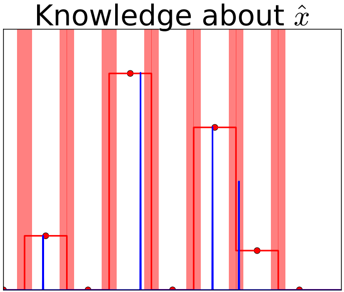


Knowledge about $\hat{x}$

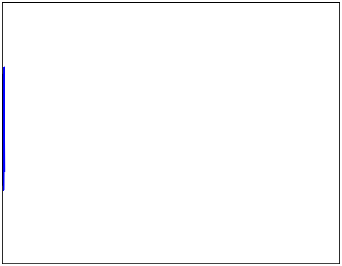


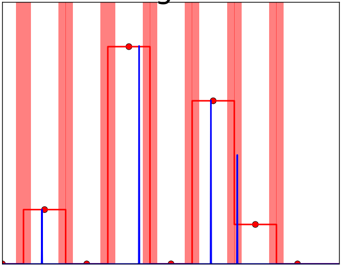
Algorithm for *exactly sparse* signals





Algorithm for *exactly sparse* signals





Lemma

If t is isolated in its bucket and in the "super-pass" region, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

Algorithm

Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

Algorithm

Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t$

Algorithm

Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t \implies$ can compute t.

Algorithm

Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t \implies$ can compute t.
  - Just like our 1-sparse recovery algorithm, $x_1/x_0 = \omega^t$.

Algorithm

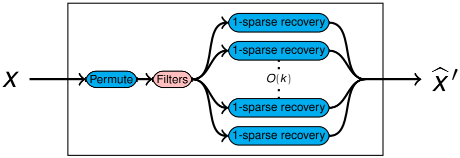
Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t \implies$ can compute t.
  - Just like our 1-sparse recovery algorithm, $x_1/x_0 = \omega^t$.
- Gives partial sparse recovery: $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.



Algorithm

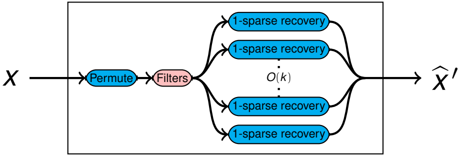
Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t \implies$ can compute t.
  - Just like our 1-sparse recovery algorithm, $x_1/x_0 = \omega^t$.
- Gives partial sparse recovery: $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.



- Repeat $k \to k/2 \to k/4 \to \cdots$

Algorithm

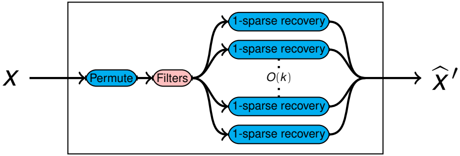
Lemma

For most t, the value b we compute for its bucket satisfies

$$b = \widehat{x}_t.$$

Computing the b for all $O(k)$ buckets takes $O(k \log n)$ time.

- Time-shift x by one and repeat: $b' = \widehat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t \implies$ can compute t.
  - Just like our 1-sparse recovery algorithm, $x_1/x_0 = \omega^t$.
- Gives partial sparse recovery: $\widehat{x}'$ such that $\widehat{x} - \widehat{x}'$ is $k/2$-sparse.



- Repeat $k \to k/2 \to k/4 \to \cdots$
- $O(k \log n)$ time sparse Fourier transform. ■

Summary (DFT setting)

- Given access to x for which $\widehat{x}$ is sparse.

Summary (DFT setting)

- Given access to x for which $\widehat{x}$ is sparse.
- Recover $\overline{x}$ such that

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

Summary (DFT setting)

- Given access to x for which $\widehat{x}$ is sparse.
- Recover $\overline{x}$ such that

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

- "Optimal" is $O(k \log(n/k))$ samples and $O(k \log(n/k) \log n)$ time

Summary (DFT setting)

- Given access to x for which $\widehat{x}$ is sparse.
- Recover $\overline{x}$ such that

$$\|\overline{x} - \widehat{x}\|_2 \leqslant (1+\epsilon) \min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$$

- "Optimal" is $O(k \log(n/k))$ samples and $O(k \log(n/k) \log n)$ time
  - Optimal samples [IK '14] OR optimal time [HIKP '12] OR $\log^c \log n$-competitive mixture [IKP '14].

Talk Outline

1. Algorithm for $k = 1$
2. Reducing k to 1
3. Putting it together
4. Continuous setting

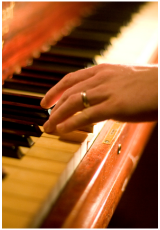
The *Continuous* Fourier Transform

Conversion between time and frequency domains

Time Domain

Fourier Transform ⟷

Frequency Domain



- The Fourier Transform $\widehat{x}$ of an integrable function $x : \mathbb{R} \to \mathbb{C}$ is

$$\widehat{x}(f) = \int_{-\infty}^{+\infty} x(t) e^{-2\pi \mathbf{i} f t} \mathrm{d}t$$

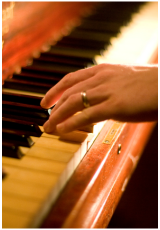
The *Continuous* Fourier Transform

Conversion between time and frequency domains

Time Domain

Frequency Domain

Fourier Transform $\longleftrightarrow$



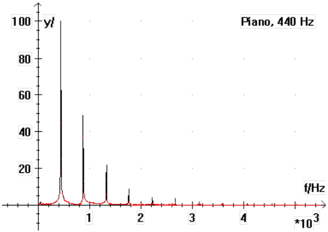
- The Fourier Transform $\widehat{x}$ of an integrable function $x : \mathbb{R} \to \mathbb{C}$ is

$$\widehat{x}(f) = \int_{-\infty}^{+\infty} x(t) e^{-2\pi \mathbf{i} f t} \mathrm{d}t$$

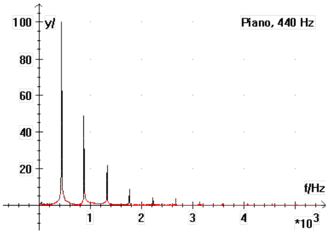
- The Inverse Transform is:

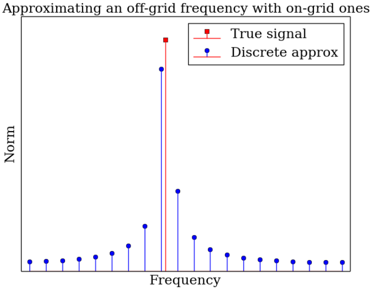
$$x(t) = \int_{-\infty}^{+\infty} \widehat{x}(f) e^{2\pi \mathbf{i} f t} \mathrm{d}f$$

Why Continuous?

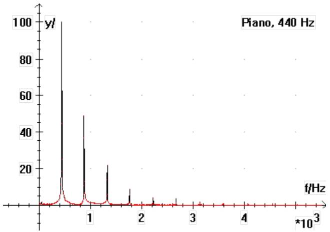


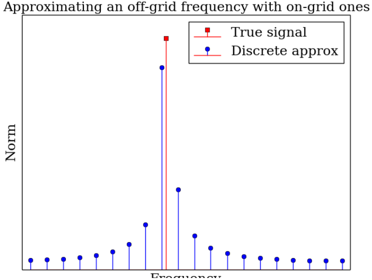
Why Continuous?

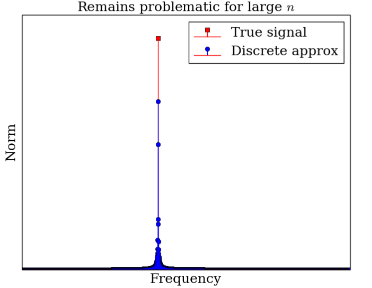




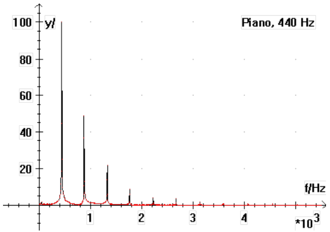
Why Continuous?

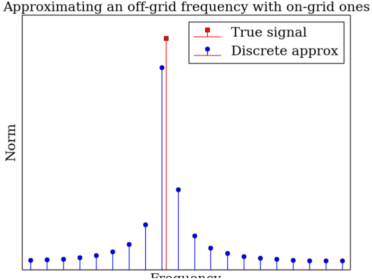


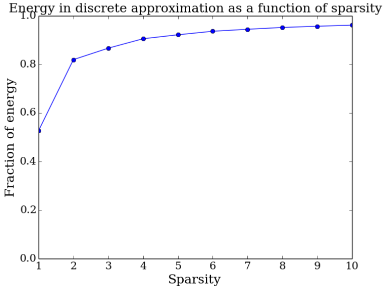


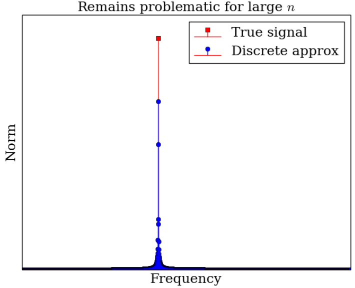


Why Continuous?



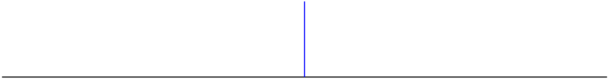




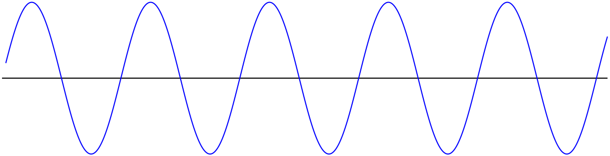


Thought Experiments

Frequency

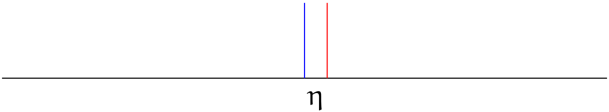


Time

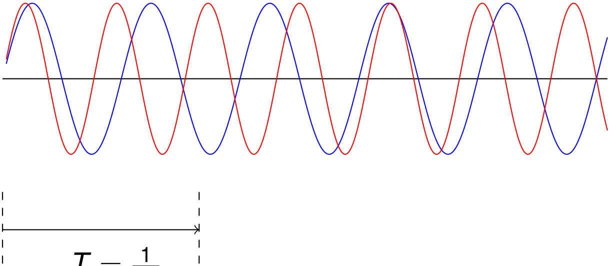


Thought Experiments

Frequency

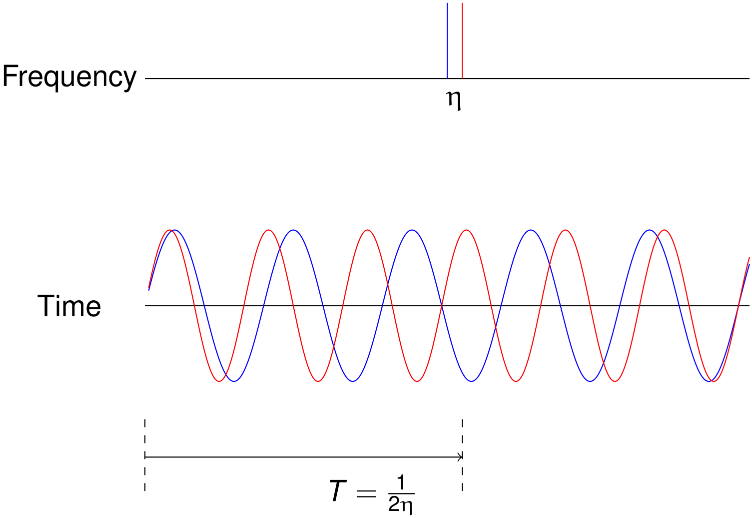


Time

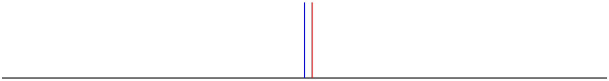


$T = \frac{1}{2\eta}$

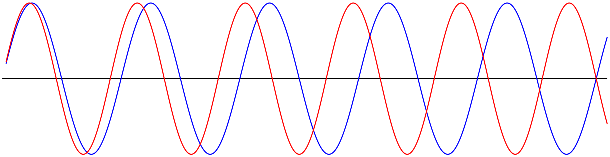
Thought Experiments



Thought Experiments

Frequency 

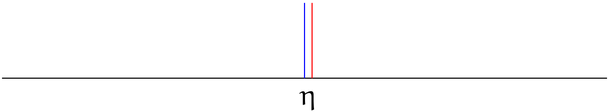
η

Time 

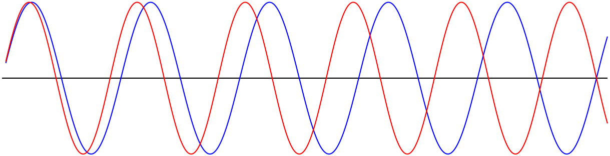
$$T = \frac{1}{2\eta}$$

Thought Experiments

Frequency



Time



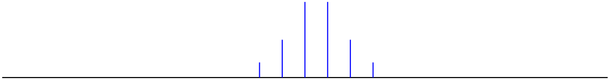
$T = \frac{1}{2\eta}$

inifinitely small η

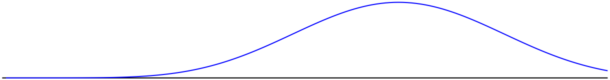
$T > 1000$ years

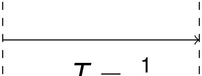
Thought Experiments 2

$$x(t) = \sin^{k-1}(\eta t)$$

Frequency  

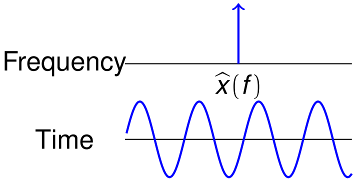
η

Time  



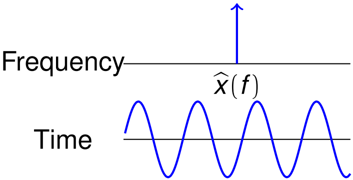
$T = \frac{1}{2\eta}$

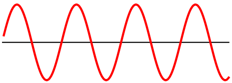
What Guarantee Do We Want?



$x(t) = e^{-2\pi \mathbf{i} f t}$

What Guarantee Do We Want?

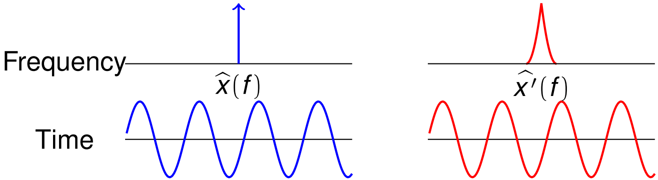




$x(t) = e^{-2\pi \mathbf{i} f t}$

$x'(t) = x(t) \cdot e^{-t/10000}$

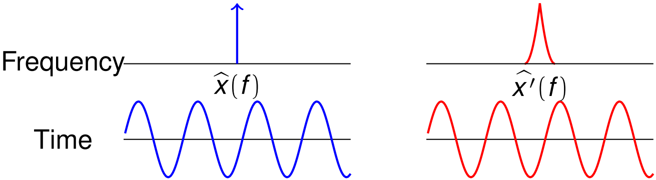
What Guarantee Do We Want?



$x(t) = e^{-2\pi \mathbf{i} f t}$

$x'(t) = x(t) \cdot e^{-t/10000}$

What Guarantee Do We Want?

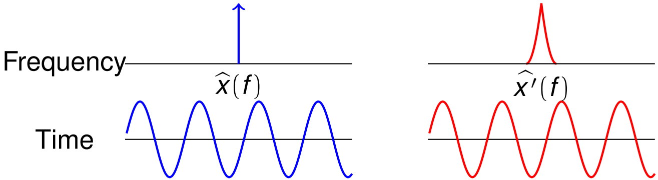


$x(t) = e^{-2\pi \mathbf{i} f t}$

$x'(t) = x(t) \cdot e^{-t/10000}$

Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse}\widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

What Guarantee Do We Want?

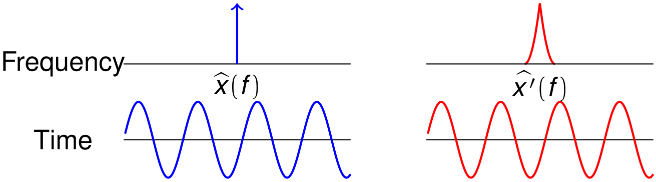


$x(t) = e^{-2\pi \mathbf{i} f t}$ $\qquad$ $x'(t) = x(t) \cdot e^{-t/10000}$

Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse}\,\widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

For red signal : $\min_{k\text{-sparse}\ \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2 = \|\widehat{x}\|_2$

What Guarantee Do We Want?



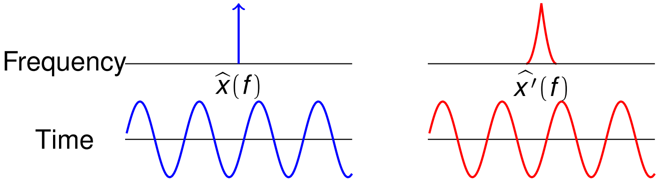
$x(t) = e^{-2\pi \mathbf{i} f t}$ $\qquad$ $x'(t) = x(t) \cdot e^{-t/10000}$

Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse}\,\widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

For red signal : $\min_{k\text{-sparse}\ \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2 = \|\widehat{x}\|_2$

DFT preserve the ℓ_2 norm

What Guarantee Do We Want?



$x(t) = e^{-2\pi \mathbf{i} ft}$

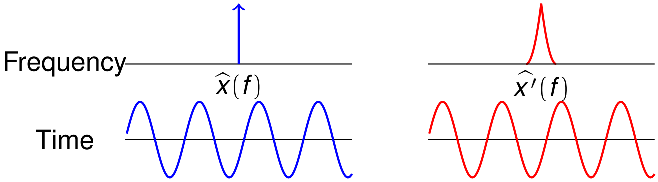
$x'(t) = x(t) \cdot e^{-t/10000}$

Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse}\widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

For red signal : $\min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2 = \|\widehat{x}\|_2$

$$\|x' - x\|_2 \lesssim \min_{k\text{-sparse } \widehat{x}_k} \|x - x_k\|_2$$

What Guarantee Do We Want?



$x(t) = e^{-2\pi \mathbf{i} f t}$

$x'(t) = x(t) \cdot e^{-t/10000}$

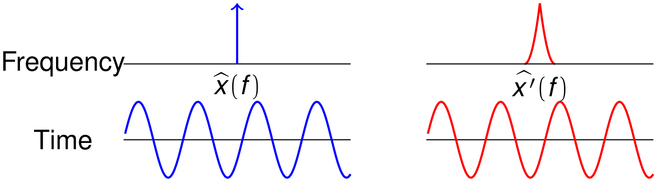
Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse} \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

For red signal : $\min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2 = \|\widehat{x}\|_2$

$$\|x' - x\|_2 \lesssim \min_{k\text{-sparse } \widehat{x}_k} \|x - x_k\|_2$$

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \min_{k\text{-sparse } \widehat{x}_k(t)} \frac{1}{T}\int_0^T |x(t) - x_k(t)|^2 \mathrm{d}t$$

What Guarantee Do We Want?



$x(t) = e^{-2\pi \mathbf{i} f t}$ $\quad$ $x'(t) = x(t) \cdot e^{-t/10000}$

Discrete FT $\|\widehat{x'} - \widehat{x}\|_2 \lesssim \min_{k\text{-sparse} \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2$

For red signal : $\min_{k\text{-sparse } \widehat{x}_k} \|\widehat{x} - \widehat{x}_k\|_2 = \|\widehat{x}\|_2$

$$\|x' - x\|_2 \lesssim \min_{k\text{-sparse } \widehat{x}_k} \|x - x_k\|_2$$

$$\boxed{\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \min_{k\text{-sparse } \widehat{x}_k(t)} \frac{1}{T}\int_0^T |x(t) - x_k(t)|^2 \mathrm{d}t}$$

Guarantee

- Sample from $x(t)$, which is approximated by a k-Fourier sparse $x_k(t)$ with η frequency separation.
- We recover an $x'(t)$ such that

$$\mathop{\mathbb{E}}_{t \in [0,T]} |x'(t) - x(t)|^2 \lesssim \mathop{\mathbb{E}}_{t \in [0,T]} |x(t) - x_k(t)|^2$$

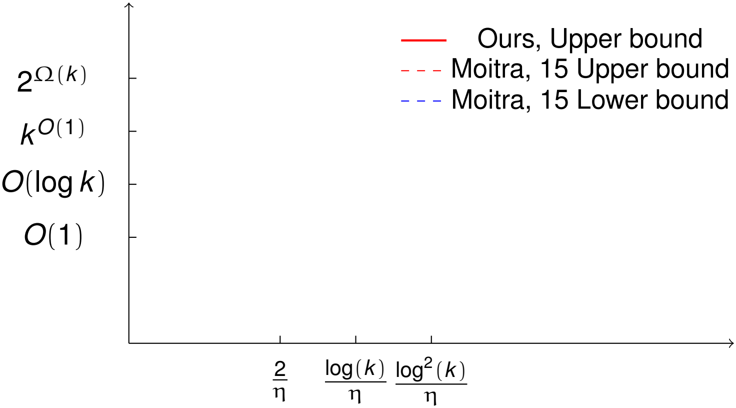
- As long as:
  - $T \geqslant O(\frac{\log^2(FT)}{\eta})$
  - Time, # samples $\geqslant O(k \log(FT) \log^2(k))$.

Previous Works and Our Results

Algorithm	Duration	Robust	Sample/Time
BCGLS, 12	$k\cdot$optimal	poor	sublinear
Moitra, 15	optimal	poly(k)	linear
Ours	$\log^2(k)\cdot$optimal	$O(1)$	sublinear

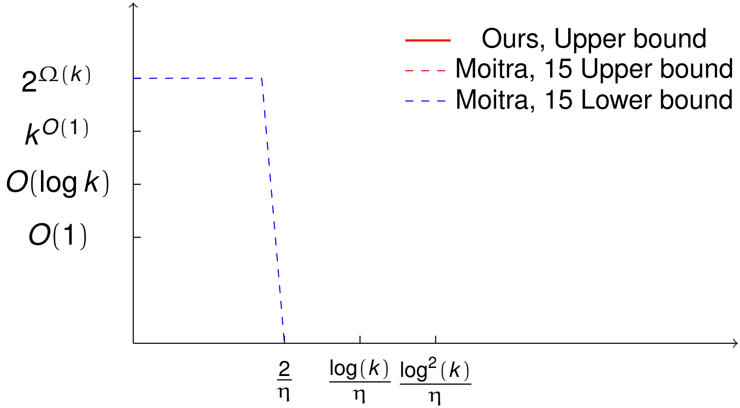
Previous Works and Our Results

Algorithm	Duration	Robust	Sample/Time
BCGLS, 12	$k \cdot$optimal	poor	sublinear
Moitra, 15	optimal	poly(k)	linear
Ours	$\log^2(k)\cdot$optimal	$O(1)$	sublinear



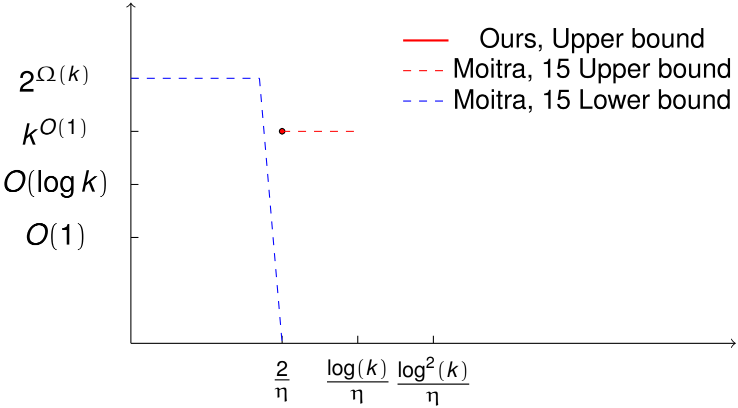
Previous Works and Our Results

Algorithm	Duration	Robust	Sample/Time
BCGLS, 12	k·optimal	poor	sublinear
Moitra, 15	optimal	poly(k)	linear
Ours	$\log^2(k)$·optimal	$O(1)$	sublinear



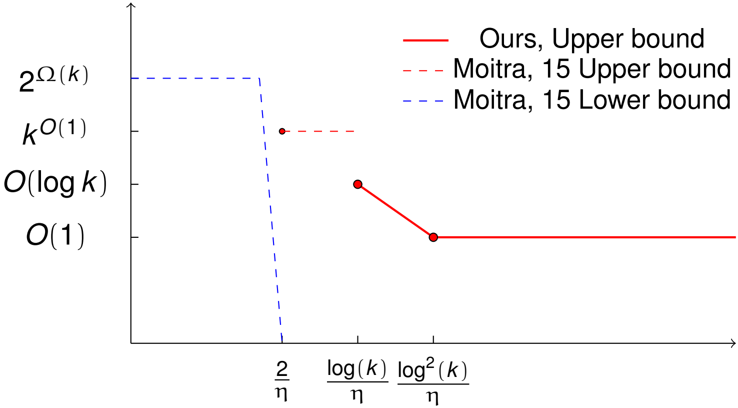
Previous Works and Our Results

Algorithm	Duration	Robust	Sample/Time
BCGLS, 12	$k\cdot$optimal	poor	sublinear
Moitra, 15	optimal	poly(k)	linear
Ours	$\log^2(k)\cdot$optimal	$O(1)$	sublinear

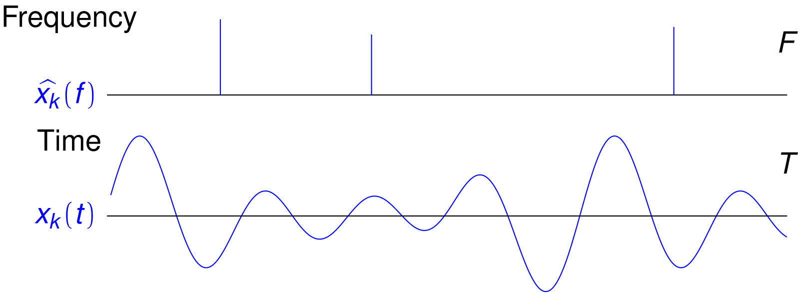


Previous Works and Our Results

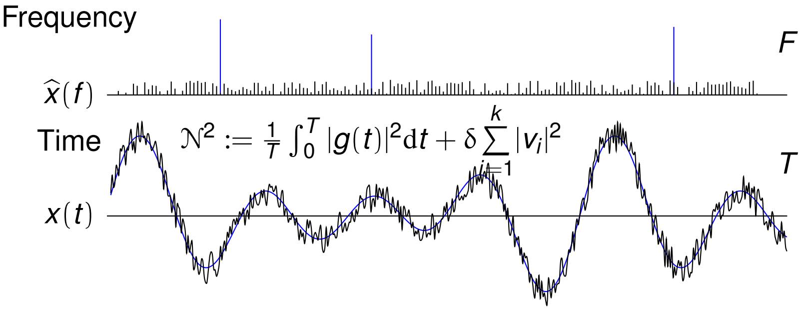
Algorithm	Duration	Robust	Sample/Time
BCGLS, 12	k·optimal	poor	sublinear
Moitra, 15	optimal	poly(k)	linear
Ours	$\log^2(k)$·optimal	$O(1)$	sublinear



Main Results

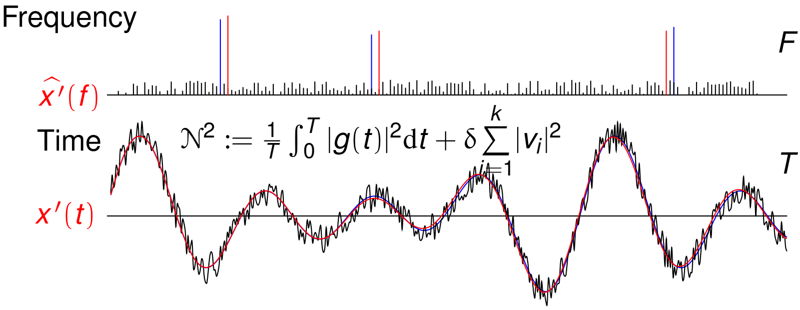


Main Results



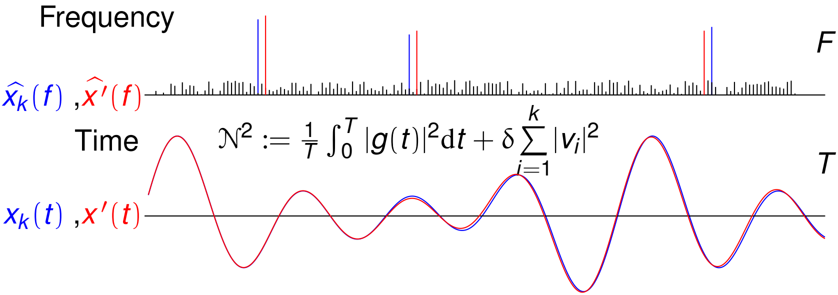
$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Main Results



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Main Results



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation

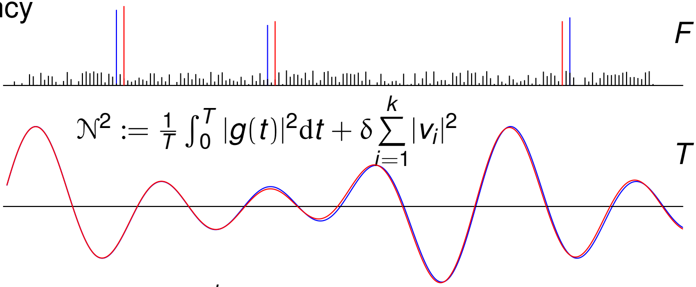
Main Results

Frequency

$\widehat{x_k}(f)$, $\widehat{x'}(f)$

Time

$x_k(t)$, $x'(t)$



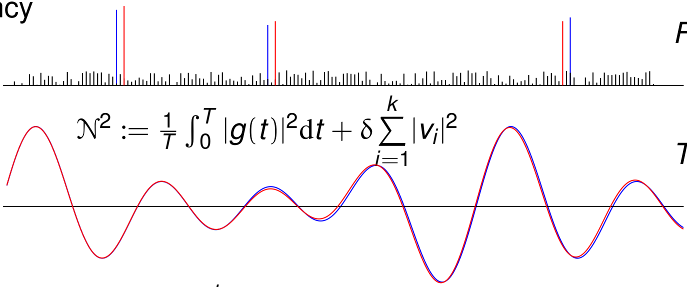
$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $$\frac{1}{T}\sum_{i=1}^{k}\int_0^T |\, v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}\,|^2 \mathrm{d}t \lesssim \mathcal{N}^2$$

Main Results

Frequency

$\widehat{x_k}(f)$, $\widehat{x'}(f)$



Time

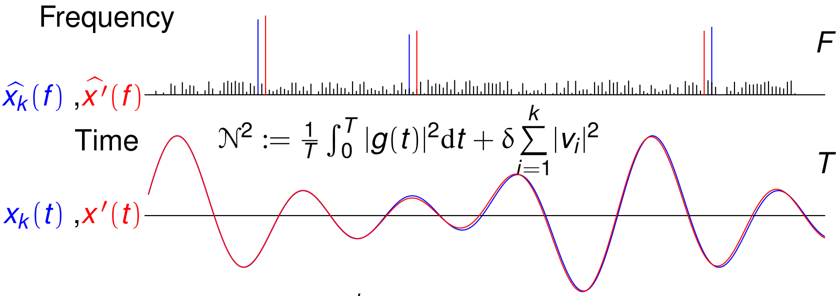
$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

$x_k(t)$, $x'(t)$

Tone Estimation $$\frac{1}{T}\sum_{i=1}^{k}\int_0^T \left| v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \right|^2 \mathrm{d}t \lesssim \mathcal{N}^2$$

Frequency Estimation

Main Results

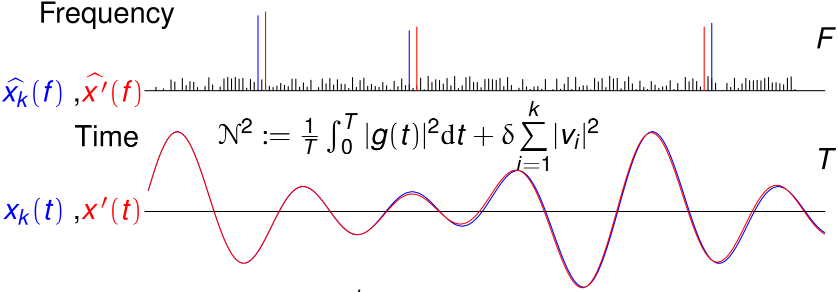


$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\frac{1}{T}\sum_{i=1}^{k}\int_0^T |\, v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}\, |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Frequency Estimation $|f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Main Results



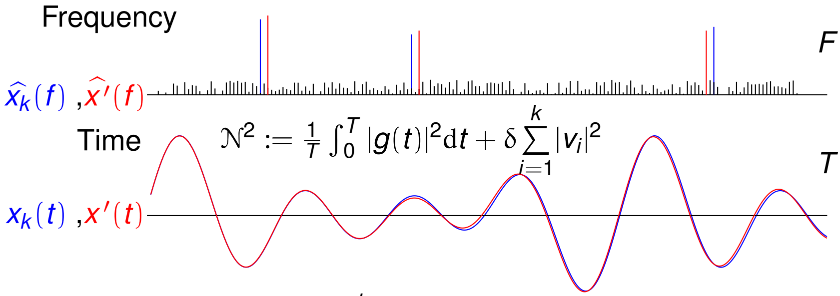
$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\frac{1}{T}\sum_{i=1}^{k} \int_0^T \left| v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \right|^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Frequency Estimation $|f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation

Main Results



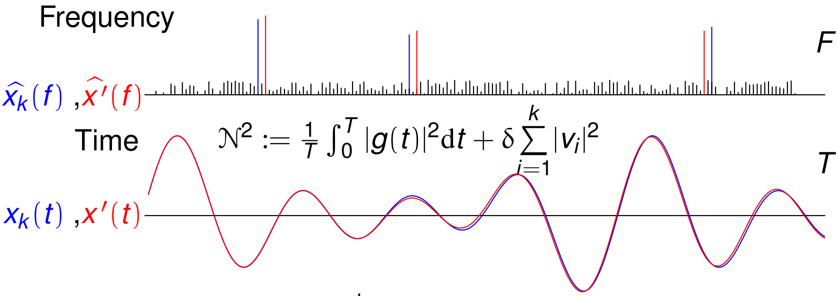
$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\frac{1}{T}\sum_{i=1}^{k}\int_0^T |\, v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \,|^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Frequency Estimation $|f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation $\frac{1}{T}\int_0^T |\sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \,|^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Main Results



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

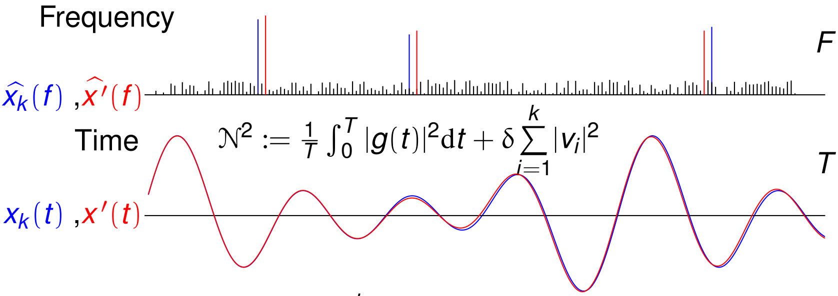
Tone Estimation $\frac{1}{T}\sum_{i=1}^{k}\int_0^T |\, v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}\, |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Frequency Estimation $|f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation $\frac{1}{T}\int_0^T |\sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}\, |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Duration

Main Results



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\frac{1}{T}\sum_{i=1}^{k} \int_0^T | v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Frequency Estimation $|f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation $\frac{1}{T}\int_0^T |\sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Duration $T = \frac{\log(k)}{\eta}$, $T = \frac{\log^2(k)}{\eta}$

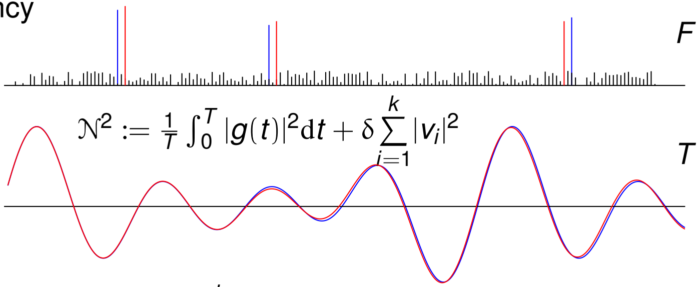
Main Results

Frequency

$\widehat{x_k}(f)$, $\widehat{x'}(f)$

Time

$x_k(t)$, $x'(t)$



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\quad \frac{1}{T}\sum_{i=1}^{k}\int_0^T | \, v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \, |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

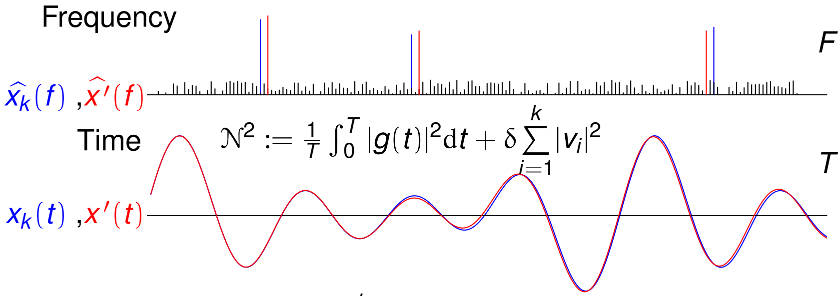
Frequency Estimation $\quad |f_i - f_i'| \lesssim \frac{1}{T\rho}$, $\rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation $\quad \frac{1}{T}\int_0^T |\sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \, |^2 \mathrm{d}t \lesssim \mathcal{N}^2$

Duration $\quad T = \frac{\log(k)}{\eta}$, $T = \frac{\log^2(k)}{\eta}$

Samples/Time

Main Results



$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^{k} |v_i|^2$$

Tone Estimation $\quad \frac{1}{T}\sum_{i=1}^{k}\int_0^T \left| v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \right|^2 \mathrm{d}t \lesssim \mathcal{N}^2$

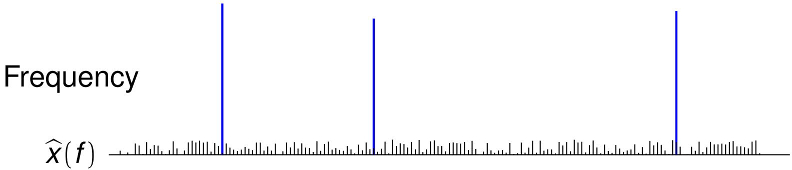
Frequency Estimation $\quad |f_i - f_i'| \lesssim \frac{1}{T\rho}, \ \rho^2 := SNR := \frac{\sum_i |v_i|^2}{\mathcal{N}^2}$

Signal Estimation $\quad \frac{1}{T}\int_0^T \left| \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t} \right|^2 \mathrm{d}t \lesssim \mathcal{N}^2$

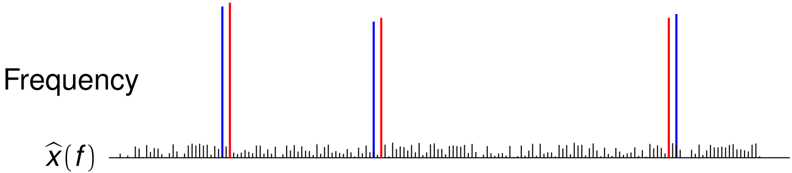
Duration $\quad T = \frac{\log(k)}{\eta}, T = \frac{\log^2(k)}{\eta}$

Samples/Time $\quad O(k \log(FT) \log^2(k))$

Tone Estimation to Signal Estimation

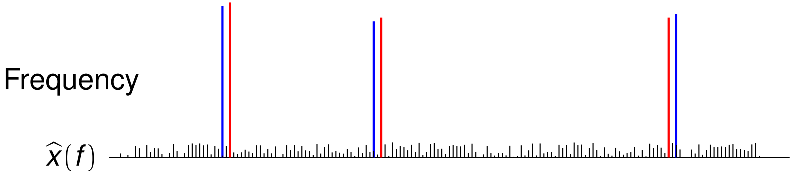


Tone Estimation to Signal Estimation



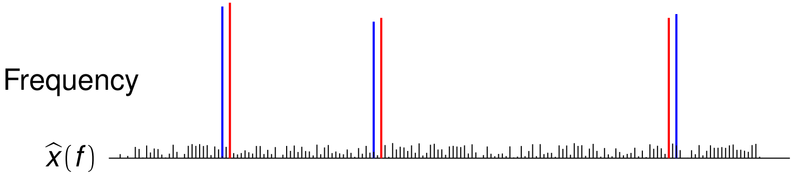
Tone Estimation to Signal Estimation

Define $\Delta_i(t) = a_i(t) - a_i'(t) \;= v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



Tone Estimation to Signal Estimation

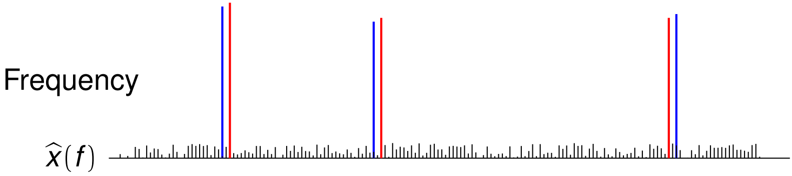
Define $\Delta_i(t) = a_i(t) - a_i'(t) \ = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



$\frac{1}{T}\int_0^T |a_1(t) - a_1'(t)|^2 \mathrm{d}t$

Tone Estimation to Signal Estimation

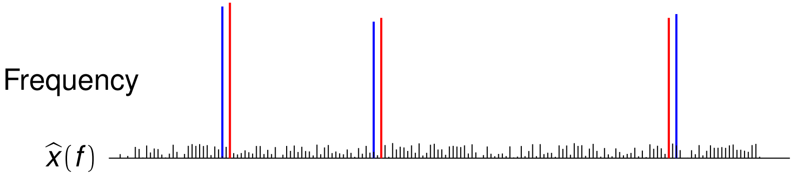
Define $\Delta_i(t) = a_i(t) - a'_i(t) = v_i e^{2\pi \mathbf{i} f_i t} - v'_i e^{2\pi \mathbf{i} f'_i t}$



$\frac{1}{T}\int_0^T |a_1(t) - a'_1(t)|^2 \mathrm{d}t \quad \frac{1}{T}\int_0^T |a_2(t) - a'_2(t)|^2 \mathrm{d}t$

Tone Estimation to Signal Estimation

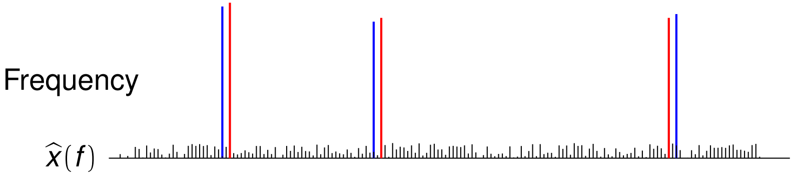
Define $\Delta_i(t) = a_i(t) - a_i'(t) \ = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



$\frac{1}{T}\int_0^T |a_1(t) - a_1'(t)|^2 \mathrm{d}t \quad \frac{1}{T}\int_0^T |a_2(t) - a_2'(t)|^2 \mathrm{d}t \quad \frac{1}{T}\int_0^T |a_3(t) - a_3'(t)|^2 \mathrm{d}t$

Tone Estimation to Signal Estimation

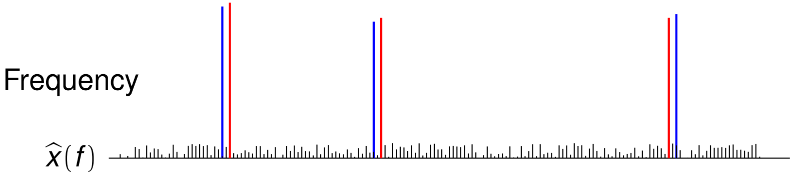
Define $\Delta_i(t) = a_i(t) - a_i'(t) = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



$\frac{1}{T}\int_0^T |a_1(t) - a_1'(t)|^2 \mathrm{d}t + \frac{1}{T}\int_0^T |a_2(t) - a_2'(t)|^2 \mathrm{d}t + \frac{1}{T}\int_0^T |a_3(t) - a_3'(t)|^2 \mathrm{d}t$

Tone Estimation to Signal Estimation

Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



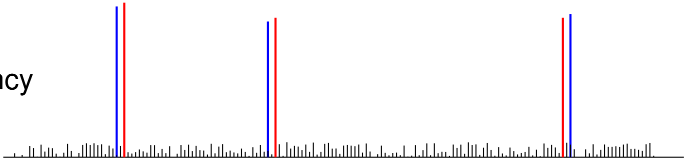
$\sum_{i=1}^{k} \frac{1}{T} \int_0^T |a_i(t)$ - $a_i'(t)|^2 \mathrm{d}t$

Tone Estimation to Signal Estimation

Define $\Delta_i(t) = a_i(t) - a_i'(t) = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Frequency

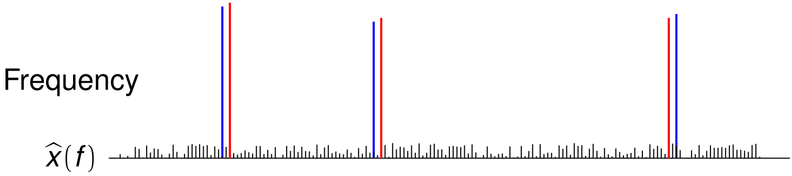
$\widehat{x}(f)$



$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t$$

Tone Estimation to Signal Estimation

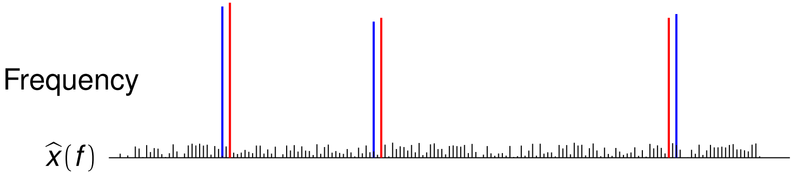
Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$



$\mathcal{N}^2 \gtrsim \sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t$ Tone Estimation

Tone Estimation to Signal Estimation

Define $\Delta_i(t) = a_i(t) - a_i'(t) = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

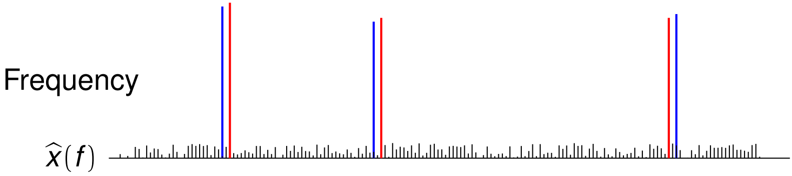


$$\mathcal{N}^2 \gtrsim \sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \quad \text{Tone Estimation}$$

$$\gtrsim \frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t \quad \text{Signal Estimation}$$

Tone Estimation to Signal Estimation

Define $\Delta_i(t) = a_i(t) - a'_i(t) \;= v_i e^{2\pi \mathbf{i} f_i t} - v'_i e^{2\pi \mathbf{i} f'_i t}$



$$\mathcal{N}^2 \gtrsim \sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \quad \text{Tone Estimation}$$

$$\gtrsim \frac{1}{T} \int_0^T \left| \sum_{i=1}^{k} \Delta_i(t) \right|^2 \mathrm{d}t \quad \text{Signal Estimation}$$

$$= \frac{1}{T} \int_0^T | x_k(t) - x'(t) |^2 \mathrm{d}t$$

Extremely Simplified Proof

$$\text{Goal}: k\sum_{i=1}^{k} y_i^2 \geqslant \left(\sum_{i=1}^{k} y_i\right)^2$$

$$\left(\sum_{i=1}^{k} y_i\right)^2$$

Extremely Simplified Proof

$$\text{Goal}: k\sum_{i=1}^{k} y_i^2 \geqslant \left(\sum_{i=1}^{k} y_i\right)^2$$

$$\left(\sum_{i=1}^{k} y_i\right)^2$$

$$= \text{diagonal terms} + \text{off-diagonal terms}$$

Extremely Simplified Proof

$$\text{Goal}: k \sum_{i=1}^{k} y_i^2 \geqslant \left(\sum_{i=1}^{k} y_i\right)^2$$

$$\left(\sum_{i=1}^{k} y_i\right)^2$$

$$= \text{diagonal terms} + \text{off-diagonal terms}$$

$$= \sum_{i=1}^{k} y_i^2 + \sum_{i \neq j} y_i y_j$$

Extremely Simplified Proof

$$\text{Goal} : k \sum_{i=1}^{k} y_i^2 \geqslant \left(\sum_{i=1}^{k} y_i\right)^2$$

$$\left(\sum_{i=1}^{k} y_i\right)^2$$

$$= \text{diagonal terms} + \text{off-diagonal terms}$$

$$= \sum_{i=1}^{k} y_i^2 + \sum_{i \neq j} y_i y_j$$

$$\leqslant \sum_{i=1}^{k} y_i^2 + \sum_{i \neq j} \frac{1}{2}(y_i^2 + y_j^2)$$

Extremely Simplified Proof

$$\text{Goal}: k \sum_{i=1}^{k} y_i^2 \geqslant \left(\sum_{i=1}^{k} y_i\right)^2$$

$$\left(\sum_{i=1}^{k} y_i\right)^2$$

$$= \text{diagonal terms} + \text{off-diagonal terms}$$

$$= \sum_{i=1}^{k} y_i^2 + \sum_{i \neq j} y_i y_j$$

$$\leqslant \sum_{i=1}^{k} y_i^2 + \sum_{i \neq j} \frac{1}{2}(y_i^2 + y_j^2)$$

$$= k \sum_{i=1}^{k} y_i^2$$

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a_i'(t) = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Goal : $\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Goal : $\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$

$$\frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$$

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Goal : $\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$

$$\frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$$

$=$ diagonal terms $+$ off-diagonal terms

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a'_i(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v'_i e^{2\pi \mathbf{i} f'_i t}$

Goal : $\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$

$$\frac{1}{T} \int_0^T |\sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$$

$=$ diagonal terms $+$ off-diagonal terms

$$= \sum_{i=1}^{k} \frac{1}{T} \int_0^T |\, \Delta_i(t) \,|^2 \mathrm{d}t + \sum_{i \neq j} \frac{1}{T} \int_0^T \Delta_i(t) \overline{\Delta_j(t)} \, \mathrm{d}t$$

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Goal : $\sum_{i=1}^k \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T | \sum_{i=1}^k \Delta_i(t)|^2 \mathrm{d}t$

$$\frac{1}{T} \int_0^T | \sum_{i=1}^k \Delta_i(t)|^2 \mathrm{d}t$$

$=$ diagonal terms $+$ off-diagonal terms

$$= \sum_{i=1}^k \frac{1}{T} \int_0^T | \Delta_i(t) |^2 \mathrm{d}t + \sum_{i \neq j} \frac{1}{T} \int_0^T \Delta_i(t) \overline{\Delta_j(t)} \, \mathrm{d}t$$

$$\lesssim (1 + \frac{\log^2(k)}{T\eta}) \cdot \sum_{i=1}^k \frac{1}{T} \int_0^T | \Delta_i(t) |^2 \mathrm{d}t$$

Simplifed Proof

$$\text{Define } \Delta_i(t) = a_i(t) - a_i'(t) \;= v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$$

$$\text{Goal : } \sum_{i=1}^k \frac{1}{T}\int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T}\int_0^T |\sum_{i=1}^k \Delta_i(t)|^2 \mathrm{d}t$$

$$\frac{1}{T}\int_0^T |\sum_{i=1}^k \Delta_i(t)|^2 \mathrm{d}t$$

$$= \text{diagonal terms} + \text{off-diagonal terms}$$

$$= \sum_{i=1}^k \frac{1}{T}\int_0^T |\,\Delta_i(t)\,|^2 \mathrm{d}t + \sum_{i \neq j} \frac{1}{T}\int_0^T \Delta_i(t)\overline{\Delta_j(t)}\,\mathrm{d}t$$

$$\lesssim \sum_{i=1}^k \frac{1}{T}\int_0^T |\,\Delta_i(t)\,|^2 \mathrm{d}t \;\text{ for } T > \log^2(k)/\eta$$

Simplifed Proof

Define $\Delta_i(t) = a_i(t) - a_i'(t) \; = v_i e^{2\pi \mathbf{i} f_i t} - v_i' e^{2\pi \mathbf{i} f_i' t}$

Goal : $\sum_{i=1}^{k} \frac{1}{T} \int_0^T |\Delta_i(t)|^2 \mathrm{d}t \gtrsim \frac{1}{T} \int_0^T | \sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$

$$\frac{1}{T} \int_0^T | \sum_{i=1}^{k} \Delta_i(t)|^2 \mathrm{d}t$$

$= \text{diagonal terms} + \text{off-diagonal terms}$

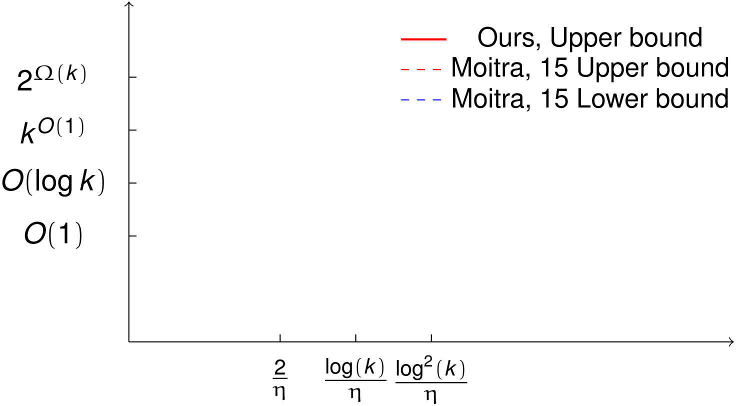
$= \sum_{i=1}^{k} \frac{1}{T} \int_0^T | \Delta_i(t) |^2 \mathrm{d}t + \sum_{i \neq j} \frac{1}{T} \int_0^T \Delta_i(t) \overline{\Delta_j(t)} \, \mathrm{d}t$

$\lesssim \quad \sum_{i=1}^{k} \frac{1}{T} \int_0^T | \Delta_i(t) |^2 \mathrm{d}t \;$ for $T > \log^2(k)/\eta$

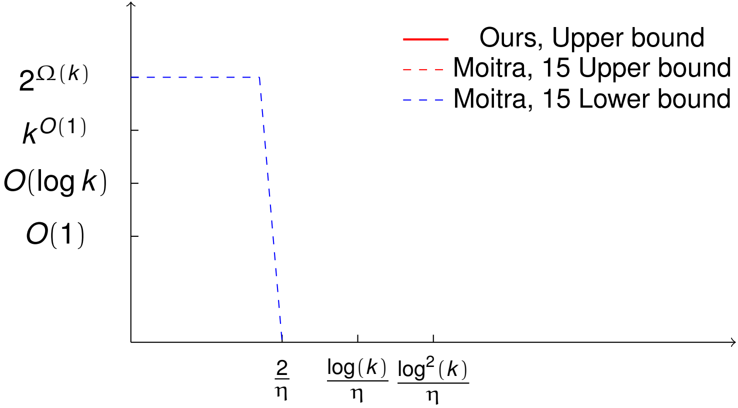
T is large enough, $\Delta_i(t)$ is more likely orthogonal to $\overline{\Delta_j(t)}$, $\forall i \neq j$

Open questions

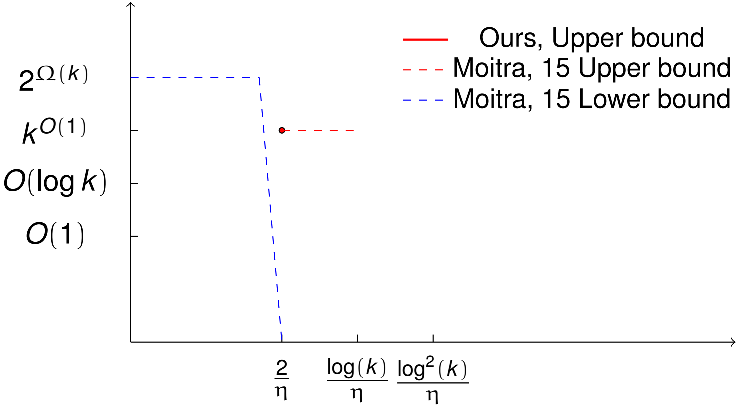
Open questions



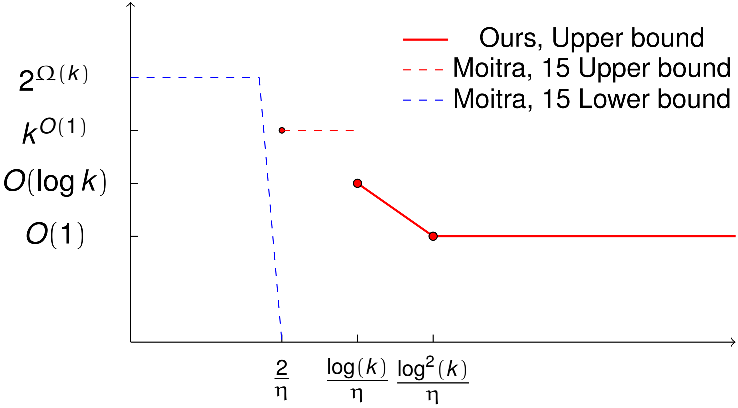
Open questions



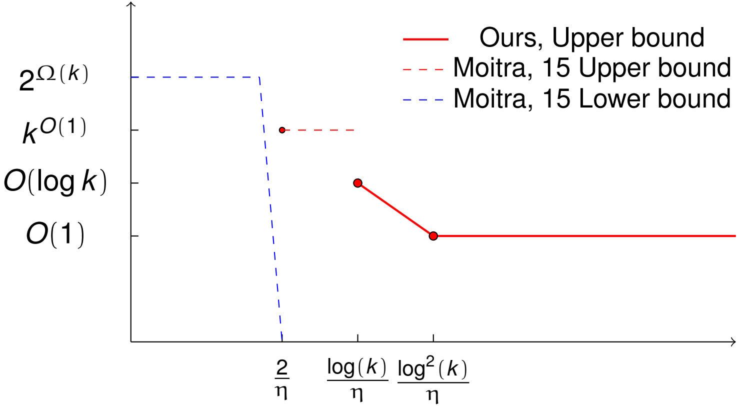
Open questions



Open questions

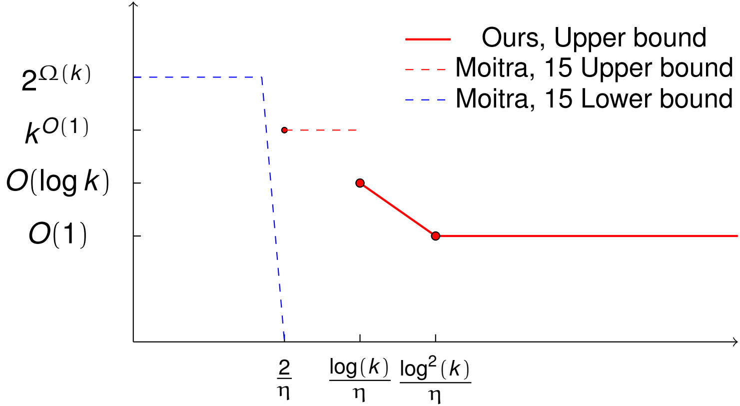


Open questions



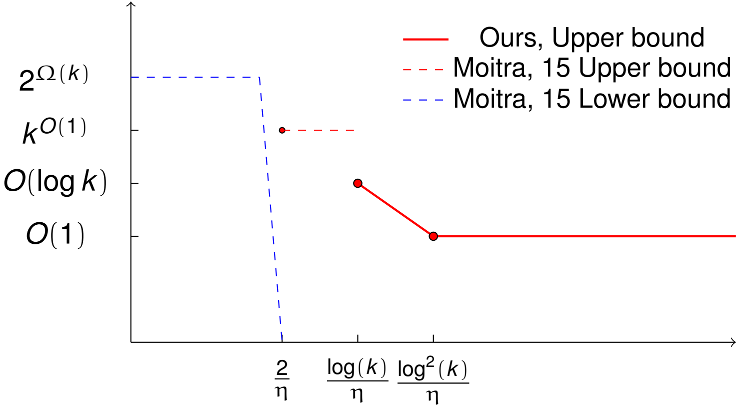
- Can we reconstruct a signal $x'(t)$ without recovering each (v_i, f_i) nicely?

Open questions



- Can we reconstruct a signal $x'(t)$ without recovering each (v_i, f_i) nicely?
- Noise is exponentially small in k, how small duration T can we pick?

Open questions



- Can we reconstruct a signal $x'(t)$ without recovering each (v_i, f_i) nicely?
- Noise is exponentially small in k, how small duration T can we pick?
- Improve our constant approximation result to $(1 \pm \epsilon)$ approximation by increasing the sample duration T?

Summary

- DFT setting: $\log^d \log n$ far from optimal in d dimensions.
- Continuous setting: more to learn.

Thank You