Sparse Fourier Transform (lecture 1)

Michael Kapralov[1]

[1] IBM Watson → EPFL

St. Petersburg CS Club
November 2015

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij},$$

where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij},$$

where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

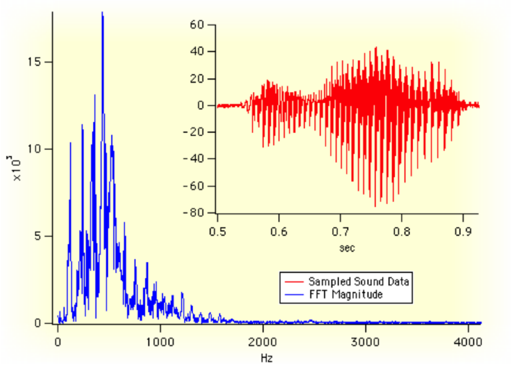
Assume that n is a power of 2.

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij},$$

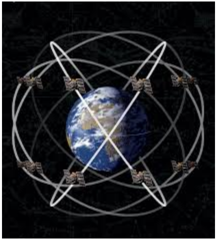
where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

Assume that n is a power of 2.



compression schemes (JPEG, MPEG)
signal processing
data analysis
imaging (MRI, NMR)

DFT has numerous applications:



Fast Fourier Transform (FFT)

Computes Discrete Fourier Transform (DFT) of a length n signal in $O(n \log n)$ time

Fast Fourier Transform (FFT)

Computes Discrete Fourier Transform (DFT) of a length n signal in $O(n \log n)$ time

Cooley and Tukey, 1964

Fast Fourier Transform (FFT)

Computes Discrete Fourier Transform (DFT) of a length n signal in $O(n \log n)$ time

Cooley and Tukey, 1964

Gauss, 1805

Fast Fourier Transform (FFT)

Computes Discrete Fourier Transform (DFT) of a length n signal in $O(n \log n)$ time

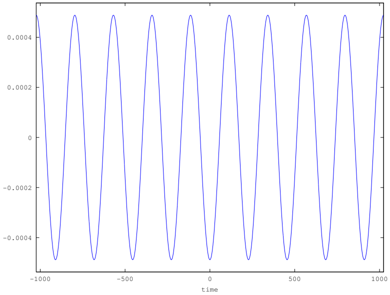
Cooley and Tukey, 1964

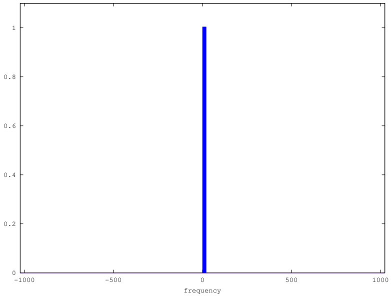
Gauss, 1805

Code=`FFTW` (Fastest Fourier Transform in the West)

Sparse FFT

Say that $\widehat{x}$ is *k*-sparse if $\widehat{x}$ has k nonzero entries

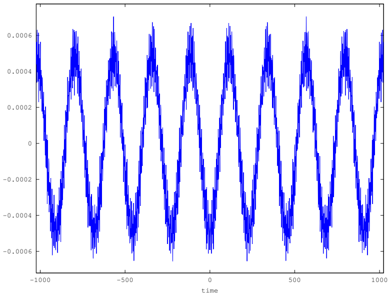


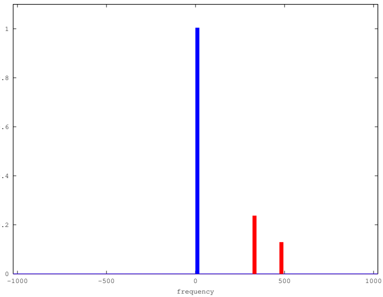


Sparse FFT

Say that $\widehat{x}$ is *k*-sparse if $\widehat{x}$ has k nonzero entries

Say that $\widehat{x}$ is approximately *k*-sparse if $\widehat{x}$ is close to k-sparse in some norm (ℓ_2 for this lecture)





Sparse approximations

Given x, compute $\widehat{x}$, then keep top k coefficients only for $k \ll N$

Used in image and video compression schemes
(e.g. JPEG, MPEG)

Sparse approximations

JPEG $\Longrightarrow$

Given x, compute $\hat{x}$, then keep top k coefficients only for $k \ll N$

Used in image and video compression schemes
(e.g. JPEG, MPEG)

Computing approximation fast

Basic approach:

- FFT computes $\widehat{x}$ from x in $O(n \log n)$ time
- compute top k coefficients in $O(n)$ time.

Computing approximation fast

Basic approach:

- FFT computes $\widehat{x}$ from x in $O(n \log n)$ time
- compute top k coefficients in $O(n)$ time.

Sparse FFT:

- directly computes k largest coefficients of $\widehat{x}$ (approximately – formal def later)
- Running time $O(k \log^2 n)$ or faster
- Sublinear time!

Sample complexity

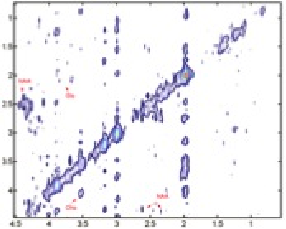
Sample complexity=number of samples accessed in time domain.

Sample complexity

In medical imaging (MRI, NMR), one measures Fourier coefficients $\widehat{x}$ of imaged object x (which is often sparse)

Sample complexity

In medical imaging (MRI, NMR), one measures Fourier coefficients $\widehat{x}$ of imaged object x (which is often sparse)



Sample complexity

Measure $\widehat{x} \in \mathbb{C}^n$, compute the Inverse Discrete Fourier Transform (IDFT) of $\widehat{x}$:

$$x_i = \sum_{j \in [n]} \widehat{x}_j \cdot \omega^{ij}.$$

Sample complexity

Measure $\widehat{x} \in \mathbb{C}^n$, compute the Inverse Discrete Fourier Transform (IDFT) of $\widehat{x}$:

$$x_i = \sum_{j \in [n]} \widehat{x}_j \cdot \omega^{ij}.$$

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij}.$$

Sample complexity

Measure $\widehat{x} \in \mathbb{C}^n$, compute the Inverse Discrete Fourier Transform (IDFT) of $\widehat{x}$:

$$x_i = \sum_{j \in [n]} \widehat{x}_j \cdot \omega^{ij}.$$

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij}.$$

Sample complexity

Measure $\widehat{x} \in \mathbb{C}^n$, compute the Inverse Discrete Fourier Transform (IDFT) of $\widehat{x}$:

$$x_i = \sum_{j \in [n]} \widehat{x}_j \cdot \omega^{ij}.$$

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform (DFT) of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-ij}.$$

Sample complexity=number of samples accessed in time domain.

Governs the measurement complexity of imaging process.

Given access to signal x in time domain, find best k-sparse approximation to $\widehat{x}$ approximately

Minimize

1. runtime
2. number of samples

Algorithms

- Randomization
- Approximation
- Hashing
- Sketching
- …

Signal processing

- Fourier transform
- Hadamard transform
- Filters
- Compressive sensing
- …

- Lecture 1: summary of techniques from
  Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02, Akavia-Goldwasser-Safra'03, Gilbert-Muthukrishnan-Strauss'05, Iwen'10, Akavia'10, Hassanieh-Indyk-Katabi-Price'12a, Hassanieh-Indyk-Katabi-Price'12b
- Lecture 2: Algorithm with $O(k \log n)$ runtime (noiseless case) Hassanieh-Indyk-Katabi-Price'12b
- Lecture 3: Algorithm with $O(k \log^2 n)$ runtime (noisy case) Hassanieh-Indyk-Katabi-Price'12b
- Lecture 4: Algorithm with $O(k \log n)$ sample complexity Indyk-Kapralov-Price'14, Indyk-Kapralov'14

Outline

1. Computing Fourier transform of 1-sparse signals fast
2. Sparsity $k > 1$: main ideas and challenges

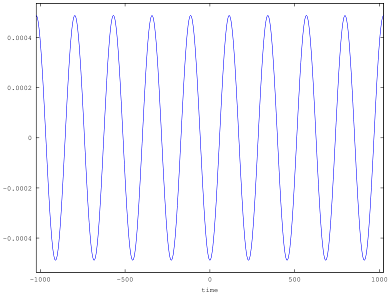
Outline

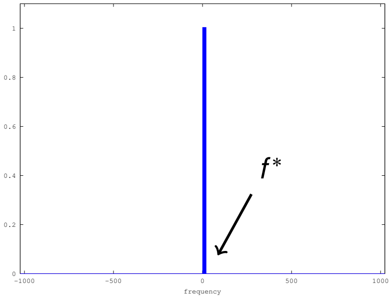
1. **Computing Fourier transform of 1-sparse signals fast**

2. Sparsity $k > 1$: main ideas and challenges

Sparse Fourier Transform ($k = 1$)

Warmup: $\widehat{x}$ is exactly 1-sparse: $\widehat{x}_f = 0$ when $f \neq f^*$ for some f^*





Note: signal is a pure frequency

Given: access to x

Need: find f^* and $\widehat{x}_{f^*}$

Two-point sampling

Input signal x is a pure frequency, so $\boxed{x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}}$

Two-point sampling

Input signal x is a pure frequency, so $\boxed{x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}}$

Sample x_0, x_1

Two-point sampling

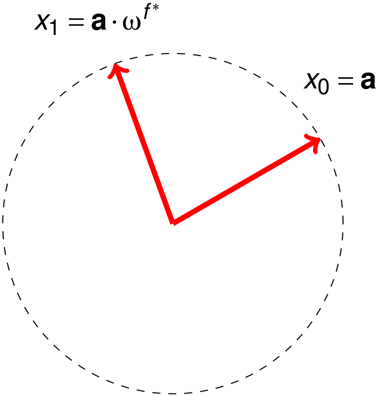
Input signal x is a pure frequency, so $\boxed{x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}}$

Sample x_0, x_1

We have

$$x_0 = \mathbf{a}$$

$$x_1 = \mathbf{a} \cdot \omega^{f^*}$$



Two-point sampling

Input signal x is a pure frequency, so $\boxed{x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}}$

Sample x_0, x_1

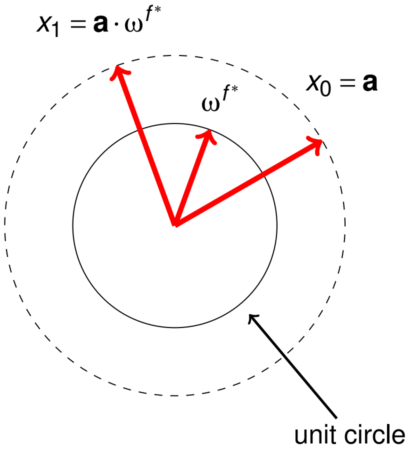
We have

$$x_0 = \mathbf{a}$$

$$x_1 = \mathbf{a} \cdot \omega^{f^*}$$

So

$$x_1 / x_0 = \omega^{f^*}$$



Two-point sampling

Input signal x is a pure frequency, so $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}$

Sample x_0, x_1

We have

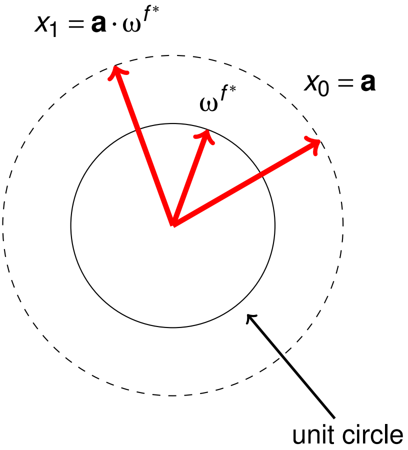
$$x_0 = \mathbf{a}$$

$$x_1 = \mathbf{a} \cdot \omega^{f^*}$$

So

$$x_1 / x_0 = \omega^{f^*}$$

Can read frequency from the angle!



Two-point sampling

Input signal x is a pure frequency, so $\boxed{x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}}$

Sample x_0, x_1

We have

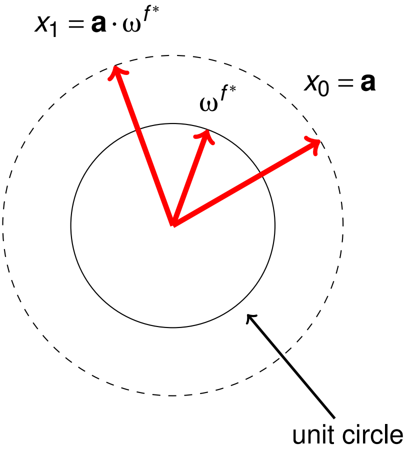
$$x_0 = \mathbf{a}$$

$$x_1 = \mathbf{a} \cdot \omega^{f^*}$$

So

$$x_1 / x_0 = \omega^{f^*}$$

Can read frequency from the angle!



Pro: constant time algorithm

Con: depends heavily on the signal being pure

Two-point sampling

Input signal x is a pure frequency **+noise**, so $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j} + \text{noise}$

Sample x_0, x_1

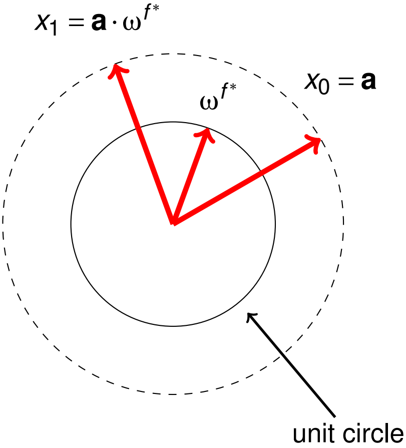
We have

$$x_0 = \mathbf{a} + \text{noise}$$
$$x_1 = \mathbf{a} \cdot \omega^{f^*} + \text{noise}$$

So

$$x_1 / x_0 = \omega^{f^*} + \text{noise}$$

Can read frequency from the angle!



Two-point sampling

Input signal x is a pure frequency**+noise**, so $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j} + \text{noise}$

Sample x_0, x_1

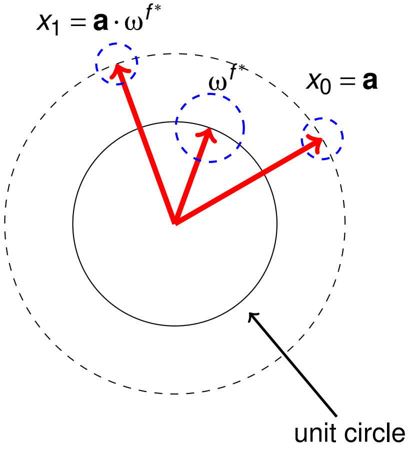
We have

$$x_0 = \mathbf{a} + \text{noise}$$
$$x_1 = \mathbf{a} \cdot \omega^{f^*} + \text{noise}$$

So

$$x_1 / x_0 = \omega^{f^*} + \text{noise}$$

Can read frequency from the angle!



Two-point sampling

Input signal x is a pure frequency**+noise**, so $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j} + \text{noise}$

Sample x_0, x_1

We have

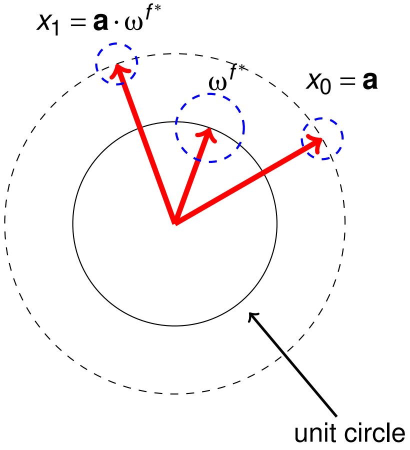
$$x_0 = \mathbf{a} + \text{noise}$$

$$x_1 = \mathbf{a} \cdot \omega^{f^*} + \text{noise}$$

So

$$x_1 / x_0 = \omega^{f^*} + \text{noise}$$

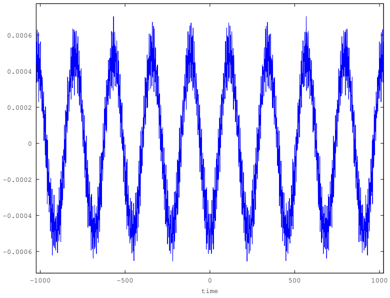
Can read frequency from the angle!

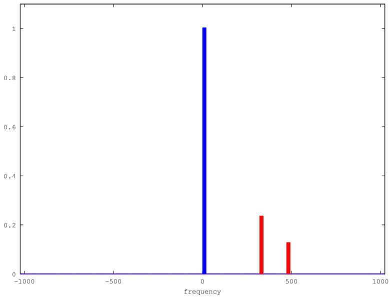


Pro: constant time algorithm
Con: depends heavily on the signal being pure

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





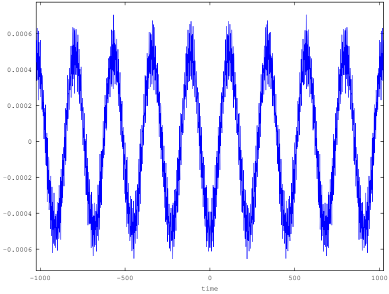
Note: signal is a pure frequency plus noise

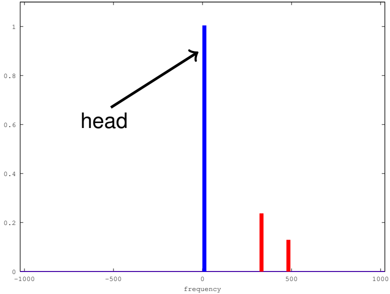
Given: access to x

Need: find f^* and $\widehat{x}_{f^*}$

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





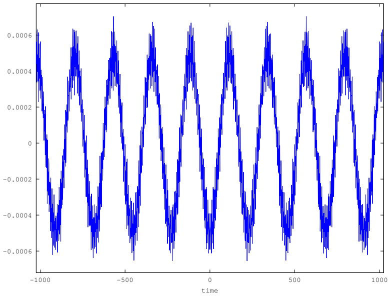
Note: signal is a pure frequency plus noise

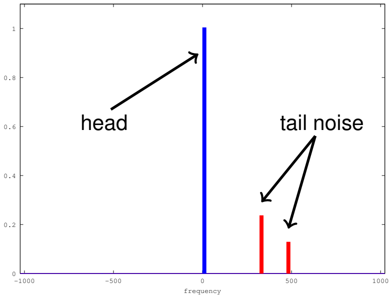
Given: access to x

Need: find f^* and $\widehat{x}_{f^*}$

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





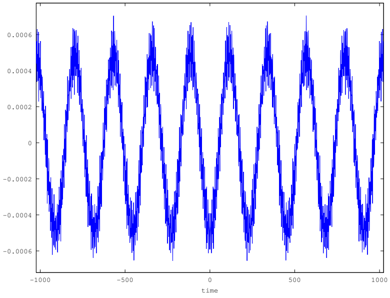
Note: signal is a pure frequency plus noise

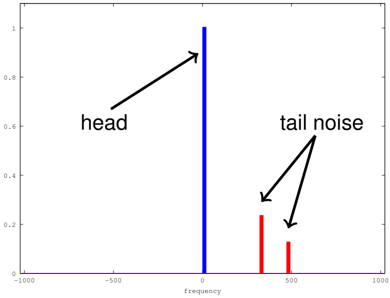
Given: access to x

Need: find f^* and $\widehat{x}_{f^*}$

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





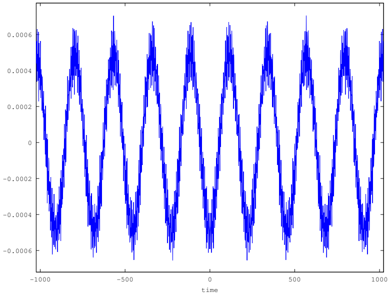
Note: signal is a pure frequency plus noise

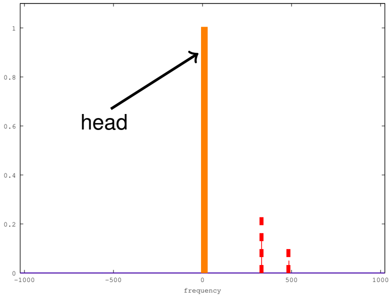
Ideally, find pure frequency $\widehat{x}'$ that approximates $\widehat{x}$ best:

$$\min_{1\text{-sparse } \widehat{x}'} ||\widehat{x} - \widehat{x}'||_2$$

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





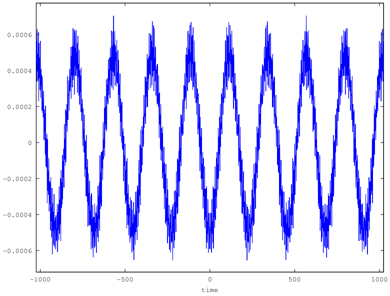
Note: signal is a pure frequency plus noise

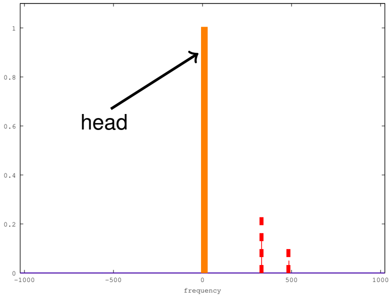
Ideally, find pure frequency $\widehat{x}'$ that approximates $\widehat{x}$ best:

$$\min_{1-\text{sparse } \widehat{x}'} ||\widehat{x} - \widehat{x}'||_2$$

Sparse Fourier Transform ($k = 1$)

Warmup – part 2: $\widehat{x}$ is 1-sparse plus noise





Note: signal is a pure frequency plus noise

Ideally, find pure frequency $\widehat{x}'$ that approximates $\widehat{x}$ best:

$$\min_{1-\text{sparse } \widehat{x}'} ||\widehat{x} - \widehat{x}'||_2 = ||\text{tail noise}||_2$$

ℓ_2/ℓ_2 sparse recovery

Ideally, find pure frequency $\widehat{x}'$ that approximates $\widehat{x}$ best

Need to allow approximation: find $\widehat{y}$ such that

$$||\widehat{x} - \widehat{y}||_2 \leq C \cdot ||\text{tail noise}||_2$$

where $C > 1$ is the approximation factor.

ℓ_2/ℓ_2 sparse recovery

Ideally, find pure frequency $\widehat{x}'$ that approximates $\widehat{x}$ best

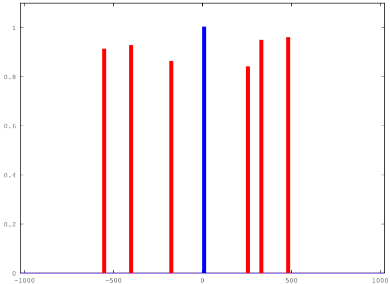
Need to allow approximation: find $\widehat{y}$ such that

$$||\widehat{x} - \widehat{y}||_2 \leq 3 \cdot ||\text{tail noise}||_2$$

Approximation guarantee

Find $\widehat{y}$ such that

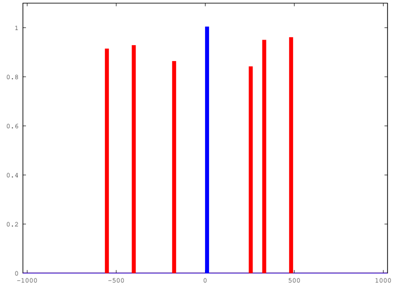
$$||\widehat{x} - \widehat{y}||_2 \leq 3 \cdot ||\text{tail noise}||_2$$



Approximation guarantee

Find $\widehat{y}$ such that

$$||\widehat{x} - \widehat{y}||_2 \leq 3 \cdot ||\text{tail noise}||_2$$



Approximation guarantee

Find $\widehat{y}$ such that

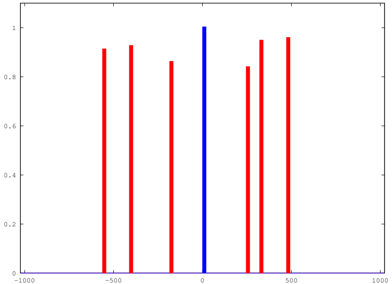
$$||\widehat{x} - \widehat{y}||_2 \leq 3 \cdot ||\text{tail noise}||_2$$

Note: only meaningful if

$$||\widehat{x}||_2 > 3 \cdot ||\text{tail noise}||_2$$

or, equivalently,

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$



Approximation guarantee

Find $\widehat{y}$ such that

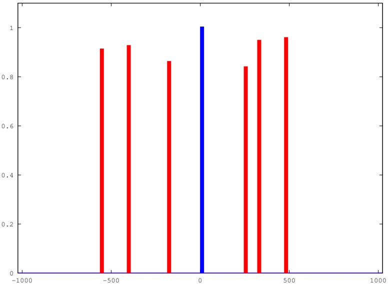
$$||\widehat{x} - \widehat{y}||_2 \leq 3 \cdot ||\text{tail noise}||_2$$

Note: only meaningful if

$$||\widehat{x}||_2 > 3 \cdot ||\text{tail noise}||_2$$

or, equivalently,

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2 \quad \text{(assume this for the lecture)}$$



Approximation guarantee

Find $\widehat{y}$ such that

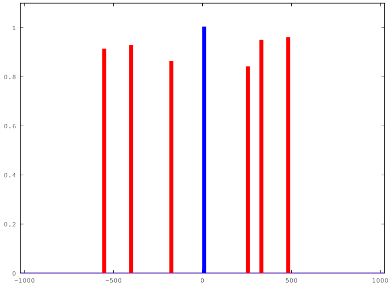
$$||\widehat{x} - \widehat{y}||_2 \le 3 \cdot ||\text{tail noise}||_2$$

Note: only meaningful if

$$||\widehat{x}||_2 > 3 \cdot ||\text{tail noise}||_2$$

or, equivalently,

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \le \varepsilon |\mathbf{a}|^2 \quad \text{(assume this for the lecture)}$$



Approximation guarantee

Find $\widehat{y}$ such that

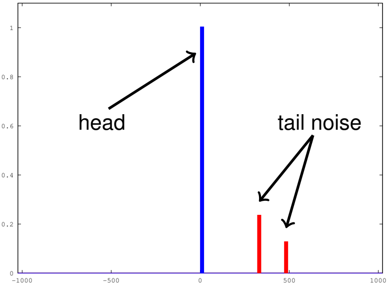
$$\|\widehat{x} - \widehat{y}\|_2 \leq 3 \cdot \|\text{tail noise}\|_2$$

Note: only meaningful if

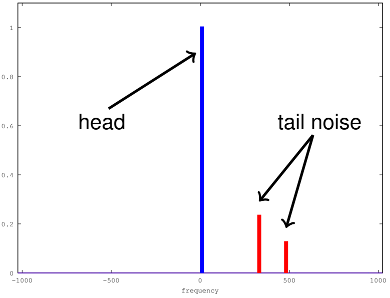
$$\|\widehat{x}\|_2 > 3 \cdot \|\text{tail noise}\|_2$$

or, equivalently,

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$ (assume this for the lecture)



A robust algorithm for finding the heavy hitter

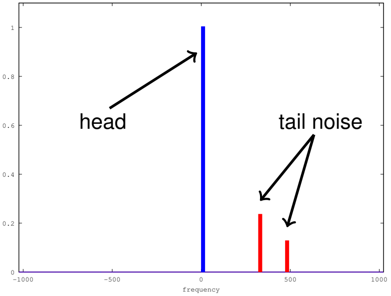


$$\text{Assume that} \quad \sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$

Describe algorithm for the noiseless case first ($\varepsilon = 0$)

Suppose that $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}$.

A robust algorithm for finding the heavy hitter



Assume that $\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$

Describe algorithm for the noiseless case first ($\varepsilon = 0$)

Suppose that $x_j = \mathbf{a} \cdot \omega^{f^* \cdot j}$.

Will find f^* bit by bit (binary search).

Bit 0

Suppose that $f^* = 2f + b$, we want b

Compute

- $x_0 = \mathbf{a}$
- $x_{n/2} = \mathbf{a} \cdot \omega^{f^* \cdot (n/2)}$

Bit 0

Suppose that $f^* = 2f + b$, we want b

Compute

- $x_0 = \mathbf{a}$
- $x_{n/2} = \mathbf{a} \cdot \omega^{f^* \cdot (n/2)}$

Claim

We have

$$x_{n/2} = x_0 \cdot (-1)^b$$

(Even frequencies are $n/2$-periodic, odd are $n/2$-antiperiodic)

Proof.

$$x_{n/2} = \mathbf{a} \cdot \omega^{f^*(n/2)} = \mathbf{a} \cdot (-1)^{2f+b} = x_0 \cdot (-1)^b$$

□

Bit 0

Suppose that $f^* = 2f + b$, we want b

Compute

- $x_{0+\mathbf{r}} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}}$
- $x_{n/2+\mathbf{r}} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})}$

Claim

For all $r \in [n]$ we have

$$x_{n/2+\mathbf{r}} = x_{0+\mathbf{r}} \cdot (-1)^b$$

(Even frequencies are $n/2$-periodic, odd are $n/2$-antiperiodic)

Proof.

$$x_{n/2+r} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}} \cdot (-1)^{2f+b} = x_{0+\mathbf{r}} \cdot (-1)^b$$

□

Bit 0

Suppose that $f^* = 2f + b$, we want b

Compute

- $x_{\mathbf{r}} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}}$
- $x_{n/2+\mathbf{r}} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})}$

Claim

For all $r \in [n]$ we have

$$x_{n/2+\mathbf{r}} = x_{\mathbf{r}} \cdot (-1)^b$$

(Even frequencies are $n/2$-periodic, odd are $n/2$-antiperiodic)

Proof.

$$x_{n/2+r} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}} \cdot (-1)^{2f+b} = x_{\mathbf{r}} \cdot (-1)^b$$

□

Bit 0

Suppose that $f^* = 2f + b$, we want b

Compute

- $x_{\mathbf{r}} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}}$
- $x_{n/2+\mathbf{r}} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})}$

Claim

For all $r \in [n]$ we have

$$x_{n/2+\mathbf{r}} = x_{\mathbf{r}} \cdot (-1)^b$$

(Even frequencies are $n/2$-periodic, odd are $n/2$-antiperiodic)

Proof.

$$x_{n/2+r} = \mathbf{a} \cdot \omega^{f^*(n/2+\mathbf{r})} = \mathbf{a} \cdot \omega^{\mathbf{f^* r}} \cdot (-1)^{2f+b} = x_{\mathbf{r}} \cdot (-1)^b$$

□

Will need arbitrary r's for the noisy setting

Bit 0 test

Set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Bit 0 test

Set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Correctness:

If $b = 0$, then $|x_{n/2+r} + x_r| = 2|x_r| = 2|\mathbf{a}|$

and $|x_{n/2+r} - x_r| = 0$

Bit 0 test

Set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Correctness:

If $b = 0$, then $|x_{n/2+r} + x_r| = 2|x_r| = 2|\mathbf{a}|$

and $|x_{n/2+r} - x_r| = 0$

If $b = 1$, then $|x_{n/2+r} + x_r| = 0$

and $|x_{n/2+r} - x_r| = 2|x_r| = 2|\mathbf{a}|$

Bit 1

Can pretend that $b_0 = 0$. Why?

Claim (Time shift theorem)

If $y_j = x_j \cdot \omega^{j \cdot \Delta}$, then $\widehat{y}_f = \widehat{x}_{f-\Delta}$.

Proof.

$$
\begin{aligned}
\widehat{y}_f &= \frac{1}{n} \sum_{j \in [n]} y_j \cdot \omega^{-fj} = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{j \cdot \Delta} \cdot \omega^{-fj} \\
&= \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-j \cdot (f - \Delta)} \\
&= \widehat{x}_{f-\Delta}
\end{aligned}
$$

□

Bit 1

Can pretend that $b_0 = 0$. Why?

Claim (Time shift theorem)

If $y_j = x_j \cdot \omega^{j \cdot \Delta}$, then $\widehat{y}_f = \widehat{x}_{f-\Delta}$.

Proof.

$$\widehat{y}_f = \frac{1}{n} \sum_{j \in [n]} y_j \cdot \omega^{-fj} = \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{j \cdot \Delta} \cdot \omega^{-fj}$$
$$= \frac{1}{n} \sum_{j \in [n]} x_j \cdot \omega^{-j \cdot (f - \Delta)}$$
$$= \widehat{x}_{f-\Delta}$$

□

If $b_0 = 1$, then replace x with $y_j := x_j \cdot \omega^{j \cdot b_0}$.

Bit 1

Assume $b_0 = 0$. Then we have $f^* = 2f$, so

$$x_j = \mathbf{a} \cdot \omega^{f^* j} = \mathbf{a} \cdot \omega^{2f \cdot j} = \mathbf{a} \cdot \omega_{N/2}^{f \cdot j}.$$

Bit 1

Assume $b_0 = 0$. Then we have $f^* = 2f$, so

$$x_j = \mathbf{a} \cdot \omega^{f^* j} = \mathbf{a} \cdot \omega^{2f \cdot j} = \mathbf{a} \cdot \omega_{N/2}^{f \cdot j}.$$

Let $\widehat{z}_j := \widehat{x}_{2j}$, i.e. spectrum of z contains even components of spectrum of $\widehat{x}$

Bit 1

Assume $b_0 = 0$. Then we have $f^* = 2f$, so

$$x_j = \mathbf{a} \cdot \omega^{f^* j} = \mathbf{a} \cdot \omega^{2f \cdot j} = \mathbf{a} \cdot \omega_{N/2}^{f \cdot j}.$$

Let $\widehat{z}_j := \widehat{x}_{2j}$, i.e. spectrum of z contains even components of spectrum of $\widehat{x}$

Then

- $(x_0, \ldots, x_{N/2-1}) = (z_0, \ldots, z_{N/2-1})$ are time samples of z_j; and
- $\widehat{z}_f = \mathbf{a}$ is the heavy hitter in z.

Bit 1

Assume $b_0 = 0$. Then we have $f^* = 2f$, so

$$x_j = \mathbf{a} \cdot \omega^{f^* j} = \mathbf{a} \cdot \omega^{2f \cdot j} = \mathbf{a} \cdot \omega_{N/2}^{f \cdot j}.$$

Let $\widehat{z}_j := \widehat{x}_{2j}$, i.e. spectrum of z contains even components of spectrum of $\widehat{x}$

Then

- $(x_0, \ldots, x_{N/2-1}) = (z_0, \ldots, z_{N/2-1})$ are time samples of z_j; and
- $\widehat{z}_f = \mathbf{a}$ is the heavy hitter in z.

So by previous derivation $z_{N/4+r} = z_r \cdot (-1)^{b_1}$

And hence

$$x_{n/4+r} \omega^{(n/4+r) b_0} = x_r \omega^{r \cdot b_0} \cdot (-1)^{b_1}$$

Bit 1

Assume $b_0 = 0$. Then we have $f^* = 2f$, so

$$x_j = \mathbf{a} \cdot \omega^{f^* j} = \mathbf{a} \cdot \omega^{2f \cdot j} = \mathbf{a} \cdot \omega_{N/2}^{f \cdot j}.$$

Let $\widehat{z}_j := \widehat{x}_{2j}$, i.e. spectrum of z contains even components of spectrum of $\widehat{x}$

Then

- $(x_0, \ldots, x_{N/2-1}) = (z_0, \ldots, z_{N/2-1})$ are time samples of z_j; and
- $\widehat{z}_f = \mathbf{a}$ is the heavy hitter in z.

So by previous derivation $z_{N/4+r} = z_r \cdot (-1)^{b_1}$

And hence

$$x_{n/4+r}\omega^{(n/4)b_0} = x_r \cdot (-1)^{b_1}$$

Decoding bit by bit

Set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Decoding bit by bit

Set $\quad b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Set $\quad b_1 \leftarrow 0$ if $|\omega^{(n/4)b_0} x_{n/4+r} + x_r| > |\omega^{(n/4)b_0} x_{n/4+r} - x_r|$

$b_1 \leftarrow 1$ o.w.

Decoding bit by bit

Set $\quad b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Set $\quad b_1 \leftarrow 0$ if $|\omega^{(n/4)b_0} x_{n/4+r} + x_r| > |\omega^{(n/4)b_0} x_{n/4+r} - x_r|$

$b_1 \leftarrow 1$ o.w.

$\ldots |\omega^{(n/8)(2b_1+b_0)} x_{n/8+r} + x_r| > |\omega^{(n/8)(2b_1+b_0)} x_{n/8+r} - x_r| \ldots$

Decoding bit by bit

Set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$

$b_0 \leftarrow 1$ o.w.

Set $b_1 \leftarrow 0$ if $|\omega^{(n/4)b_0} x_{n/4+r} + x_r| > |\omega^{(n/4)b_0} x_{n/4+r} - x_r|$

$b_1 \leftarrow 1$ o.w.

$\ldots |\omega^{(n/8)(2b_1+b_0)} x_{n/8+r} + x_r| > |\omega^{(n/8)(2b_1+b_0)} x_{n/8+r} - x_r| \ldots$

Overall: $O(\log n)$ samples to identify f^*. Runtime $O(\log n)$

Noisy setting (dealing with ε)

We now have

$$x_j = \mathbf{a} \cdot \omega^{f^* \cdot j} + \sum_{f \neq f^*} \widehat{x}_f \omega^{fj}$$

$$= \mathbf{a} \cdot \omega^{f^* \cdot j} + \mu_j \quad (\mu_j \text{ is the noise in time domain})$$

Argue that μ_j is usually small?

Noisy setting (dealing with ε)

We now have

$$x_j = \mathbf{a} \cdot \omega^{f^* \cdot j} + \sum_{f \neq f^*} \widehat{x}_f \omega^{fj}$$
$$= \mathbf{a} \cdot \omega^{f^* \cdot j} + \mu_j \quad (\mu_j \text{ is the noise in time domain})$$

Argue that μ_j is usually small?

Parseval's equality: noise energy in time domain is proportional to noise energy in frequency domain:

$$\sum_{j=0}^{N-1} |\mu_j|^2 = n \sum_{f \neq f^*} |\widehat{x}_f|^2.$$

Noisy setting (dealing with ε)

We now have

$$\begin{aligned} x_j &= \mathbf{a} \cdot \omega^{f^* \cdot j} + \sum_{f \neq f^*} \widehat{x}_f \omega^{fj} \\ &= \mathbf{a} \cdot \omega^{f^* \cdot j} + \mu_j \quad (\mu_j \text{ is the noise in time domain}) \end{aligned}$$

Argue that μ_j is usually small?

Parseval's equality: noise energy in time domain is proportional to noise energy in frequency domain:

$$\sum_{j=0}^{N-1} |\mu_j|^2 = n \sum_{f \neq f^*} |\widehat{x}_f|^2.$$

So on average $|\mu_j|^2$ is small:

$$\mathbf{E}_j[|\mu_j|^2] \leq \sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$

Need to ensure that:

1. f^* is decoded correctly

2. **a** is estimated well enough to satisfy ℓ_2/ℓ_2 guarantees:

$$||\widehat{x} - \widehat{y}||_2 \le C \cdot ||\widehat{x} - \widehat{x}'||_2$$

Decoding in the noisy setting

Bit 0: set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$ and $b_0 \leftarrow 1$ o.w.

Claim

If $\mu_{n/2+r} < |\mathbf{a}|/2$ *and* $\mu_r < |\mathbf{a}|/2$*, then outcome of the bit test is the same.*

Decoding in the noisy setting

Bit 0: set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$ and $b_0 \leftarrow 1$ o.w.

Claim

If $\mu_{n/2+r} < |\mathbf{a}|/2$ *and* $\mu_r < |\mathbf{a}|/2$*, then outcome of the bit test is the same.*

Suppose $b_0 = 1$.

Decoding in the noisy setting

Bit 0: set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$ and $b_0 \leftarrow 1$ o.w.

Claim

If $\mu_{n/2+r} < |\mathbf{a}|/2$ *and* $\mu_r < |\mathbf{a}|/2$*, then outcome of the bit test is the same.*

Suppose $b_0 = 1$.

Then

$$|x_{n/2+r} + x_r| \leq |\mu_{n/2+r}| + |\mu_r| < |\mathbf{a}|$$

Decoding in the noisy setting

Bit 0: set $b_0 \leftarrow 0$ if $|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$ and $b_0 \leftarrow 1$ o.w.

Claim

If $\mu_{n/2+r} < |\mathbf{a}|/2$ and $\mu_r < |\mathbf{a}|/2$, then outcome of the bit test is the same.

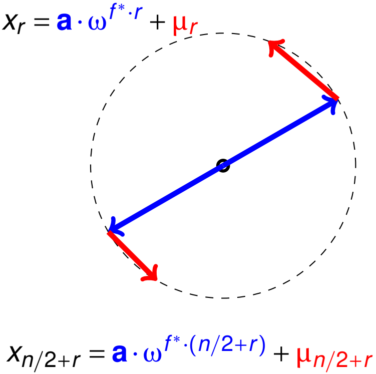
Suppose $b_0 = 1$.

Then

$$|x_{n/2+r} + x_r| \leq |\mu_{n/2+r}| + |\mu_r| < |\mathbf{a}|$$

and

$$|x_{n/2+r} - x_r| \geq 2|\mathbf{a}| - |\mu_{n/2+r}| - |\mu_r| > |\mathbf{a}|$$



Decoding in the noisy setting

On average $|\mu_j|^2$ is small:

$$\mathbf{E}_j[|\mu_j|^2] \leq \sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$

By Markov's inequality

$$\mathbf{Pr}_j[|\mu_j|^2 > |\mathbf{a}|^2/4] \leq \mathbf{Pr}_j[|\mu_j|^2 > (1/(4\varepsilon)) \cdot \mathbf{E}_j[|\mu_j|^2]] \leq 4\varepsilon$$

Decoding in the noisy setting

On average $|\mu_j|^2$ is small:

$$\mathbf{E}_j[|\mu_j|^2] \leq \sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\mathbf{a}|^2$$

By Markov's inequality

$$\mathbf{Pr}_j[|\mu_j|^2 > |\mathbf{a}|^2/4] \leq \mathbf{Pr}_j[|\mu_j|^2 > (1/(4\varepsilon)) \cdot \mathbf{E}_j[|\mu_j|^2]] \leq 4\varepsilon$$

By a union bound

$$\mathbf{Pr}_r[|\mu_r| \leq |\mathbf{a}|/2 \text{ and } |\mu_{n/2+r}| \leq |\mathbf{a}|/2] \geq 1 - 8\varepsilon$$

Decoding in the noisy setting

On average $|\mu_j|^2$ is small:

$$\mathbf{E}_j[|\mu_j|^2] \le \sum_{f \neq f^*} |\widehat{x}_f|^2 \le \varepsilon |\mathbf{a}|^2$$

By Markov's inequality

$$\mathbf{Pr}_j[|\mu_j|^2 > |\mathbf{a}|^2/4] \le \mathbf{Pr}_j[|\mu_j|^2 > (\mathbf{1}/(\mathbf{4}\varepsilon)) \cdot \mathbf{E}_j[|\mu_j|^2]] \le \mathbf{4}\varepsilon$$

By a union bound

$$\mathbf{Pr}_r[|\mu_r| \le |\mathbf{a}|/2 \text{ and } |\mu_{n/2+r}| \le |\mathbf{a}|/2] \ge 1 - \mathbf{8}\varepsilon$$

Thus, a bit test is correct with probability at least $1 - \mathbf{8}\varepsilon$.

Decoding in the noisy setting

Bit 0: set b_0 to zero if

$$|x_{n/2+r} + x_r| > |x_{n/2+r} - x_r|$$

and to 1 otherwise

For $\varepsilon < 1/64$ each test is correct with probability $\geq \mathbf{3/4}$.

Final test: perform $T \gg 1$ independent tests, use majority vote.

How large should T be? Success probability?

Decoding in the noisy setting

For $j = 1, \ldots, T$ let

$$Z_j = \begin{cases} 1 & \text{if } j\text{-th test is correct} \\ 0 & \text{o.w.} \end{cases}$$

We have $\mathbf{E}[Z_j] \geq 3/4$.

Decoding in the noisy setting

For $j = 1, \ldots, T$ let

$$Z_j = \begin{cases} 1 & \text{if } j\text{-th test is correct} \\ 0 & \text{o.w.} \end{cases}$$

We have $\mathbf{E}[Z_j] \geq 3/4$.

Chernoff bounds

$$\mathbf{Pr}[\sum_{j=1}^{T} Z_j < T/2] < e^{-\Omega(T)}.$$

Set $T = O(\log \log n)$

Majority is correct with probability at least $1 - 1/(16 \log_2 n)$

So all bits correct with probability $\geq 15/16$

Estimating the value of heavy hitter

Recall that

$$x_r = \mathbf{a} \cdot \omega^{f^* \cdot r} + \mu_r \quad (noise)$$

Our estimate: pick random $r \in [n]$ and output

$$\text{est} \leftarrow \mathbf{x_r} \omega^{-\mathbf{f^* \cdot r}}$$

Estimating the value of heavy hitter

Recall that

$$x_r = \mathbf{a} \cdot \omega^{f^* \cdot r} + \mu_r \quad (noise)$$

Our estimate: pick random $r \in [n]$ and output

$$\text{est} \leftarrow \mathbf{x_r} \omega^{-\mathbf{f^* \cdot r}}$$

Expected squared error?

$\mathbf{E}_r[|est - \mathbf{a}|^2]$

Estimating the value of heavy hitter

Recall that

$$x_r = \mathbf{a} \cdot \omega^{f^* \cdot r} + \mu_r \quad (noise)$$

Our estimate: pick random $r \in [n]$ and output

$$\text{est} \leftarrow \mathbf{x_r} \omega^{-\mathbf{f}^* \cdot \mathbf{r}}$$

Expected squared error?

$$\mathbf{E}_r[|est - \mathbf{a}|^2] = \mathbf{E}_r[|x_r \omega^{-f^* \cdot r} - \mathbf{a}|^2]$$

Estimating the value of heavy hitter

Recall that

$$x_r = \mathbf{a} \cdot \omega^{f^* \cdot r} + \mu_r \quad (\textit{noise})$$

Our estimate: pick random $r \in [n]$ and output

$$\text{est} \leftarrow \mathbf{x_r} \omega^{-\mathbf{f^* \cdot r}}$$

Expected squared error?

$$\mathbf{E}_r[|est - \mathbf{a}|^2] = \mathbf{E}_r[|x_r \omega^{-f^* \cdot r} - \mathbf{a}|^2] = \mathbf{E}_r[|x_r - \mathbf{a} \cdot \omega^{f^* \cdot r}|^2]$$

Estimating the value of heavy hitter

Recall that

$$x_r = \mathbf{a} \cdot \omega^{f^* \cdot r} + \mu_r \quad (noise)$$

Our estimate: pick random $r \in [n]$ and output

$$\text{est} \leftarrow \mathbf{x_r} \omega^{-\mathbf{f}^* \cdot \mathbf{r}}$$

Expected squared error?

$$\mathbf{E}_r[|est - \mathbf{a}|^2] = \mathbf{E}_r[|x_r \omega^{-f^* \cdot r} - \mathbf{a}|^2] = \mathbf{E}_r[|x_r - \mathbf{a} \cdot \omega^{f^* \cdot r}|^2] = \mathbf{E}_r[|\mu_r|^2]$$

Now by Markov's inequality

$$\mathbf{Pr}_r[|est - \mathbf{a}|^2 > 4\varepsilon|\mathbf{a}|^2] < 1/4.$$

Putting it together: algorithm for 1-sparse signals

Let

$$\widehat{y}_f = \begin{cases} est & \text{if } f = f^* \\ 0 & \text{o.w.} \end{cases}$$

By triangle inequality

$$\begin{aligned} ||\widehat{y} - \widehat{x}||_2 &\le ||\widehat{y}_{f^*} - \mathbf{a}||_2 + ||\widehat{y}_{-f^*} - \widehat{x}_{-f^*}||_2 \\ &\le 2\sqrt{\varepsilon}|\mathbf{a}| + \sqrt{\varepsilon}|\mathbf{a}| \\ &= 3||\widehat{x} - \widehat{x}'||_2. \end{aligned}$$

Thus, with probability $\ge 2/3$ our algorithm satisfies ℓ_2/ℓ_2 guarantee with $C = 3$.

Runtime = $O(\log n \log\log n)$

Sample complexity = $O(\log n \log\log n)$

Runtime=$O(\log n \log\log n)$

Sample complexity=$O(\log n \log\log n)$

Ex. 1: reduce sample complexity to $O(\log n)$, keep $O(\text{poly}(\log n))$ runtime

Ex. 2: reduce sample complexity to $O(\log_{1/\varepsilon} n)$

Runtime=$O(\log n \log\log n)$

Sample complexity=$O(\log n \log\log n)$

Ex. 1: reduce sample complexity to $O(\log n)$, keep $O(\mathrm{poly}(\log n))$ runtime

Ex. 2: reduce sample complexity to $O(\log_{1/\varepsilon} n)$

What about $k > 1$

Outline

1. Sparsity: definitions, motivation
2. Computing Fourier transform of 1-sparse signals fast
3. **Sparsity $k > 1$: main ideas and challenges**

Sparsity $k > 1$

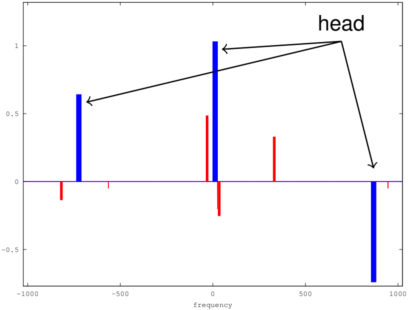
Let $\widehat{x}' \leftarrow$ best k-sparse approximation of $\widehat{x}$

Our goal: find $\widehat{y}$ such that

$$||\widehat{x} - \widehat{y}||_2 \leq C \cdot ||\widehat{x} - \widehat{x}'||_2$$

where $C > 1$ is the approximation factor.

(This is the ℓ_2/ℓ_2 guarantee)



Sparsity $k > 1$

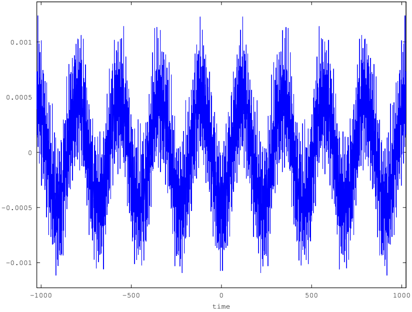
Main idea: implement hashing to reduce to 1-sparse case:

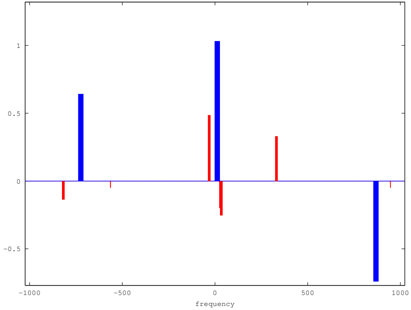
- 'hash' frequencies into $\approx k$ bins
- run 1-sparse algo on isolated elements

Assumption: can randomly permute frequencies (will remove in next lecture)

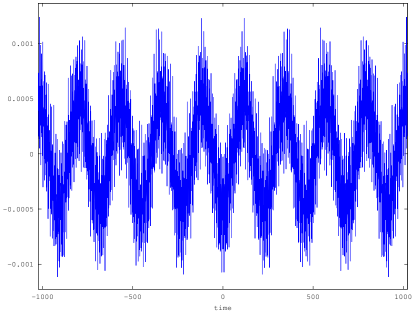
Implement hashing? Need to design a bucketing scheme for the frequency domain

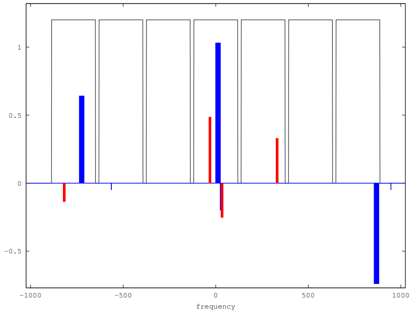
Partition frequency domain into $B \approx k$ buckets



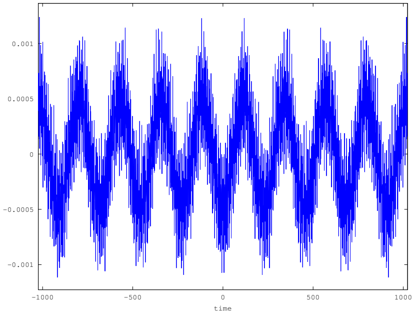


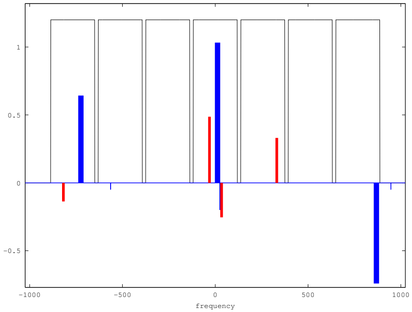
Partition frequency domain into $B \approx k$ buckets





Partition frequency domain into $B \approx k$ buckets



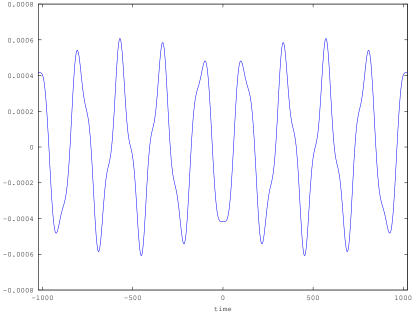


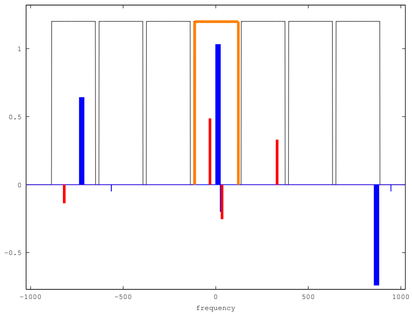
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



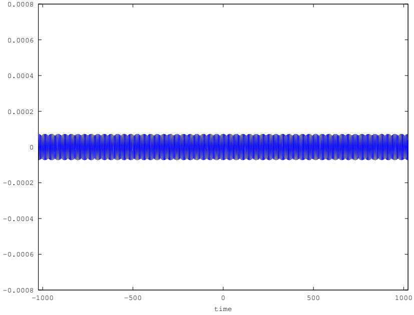


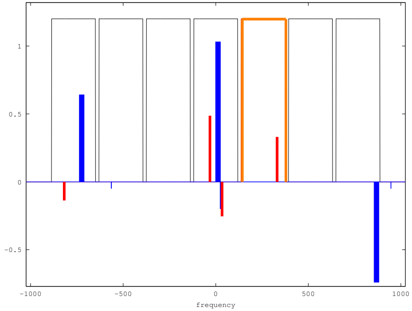
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



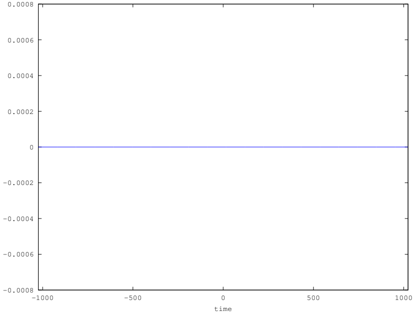


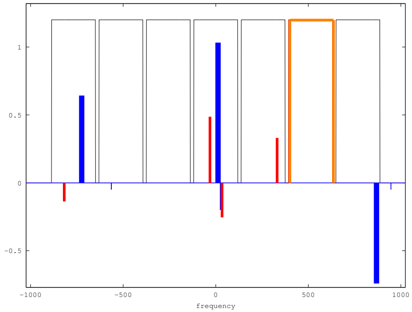
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



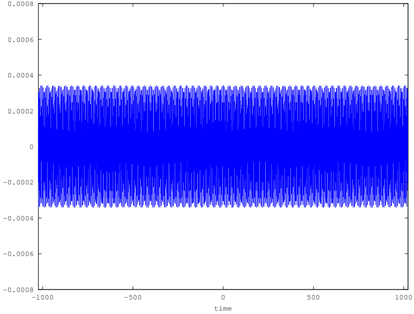


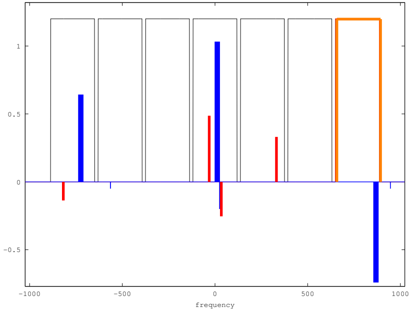
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



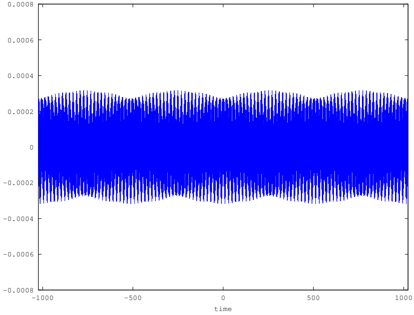


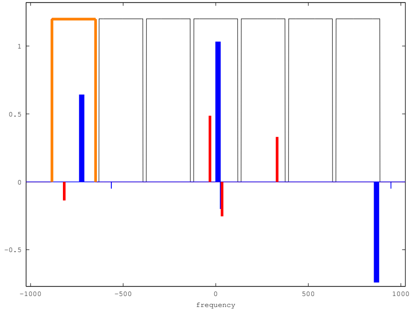
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



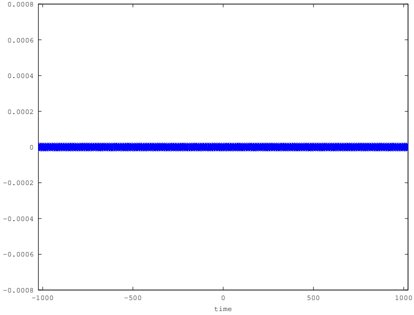


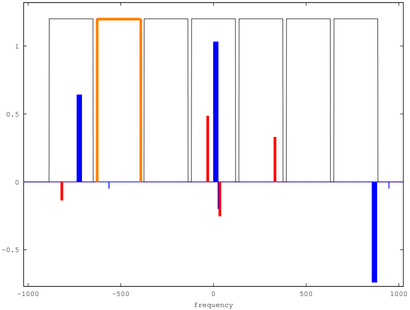
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



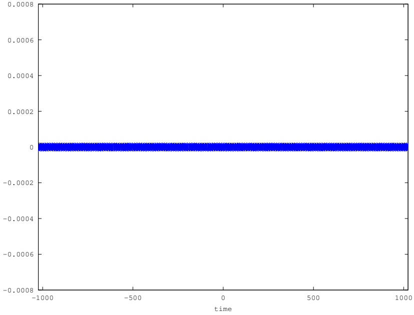


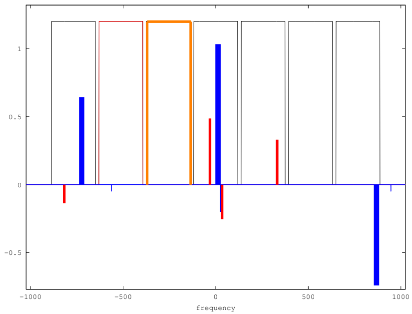
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets





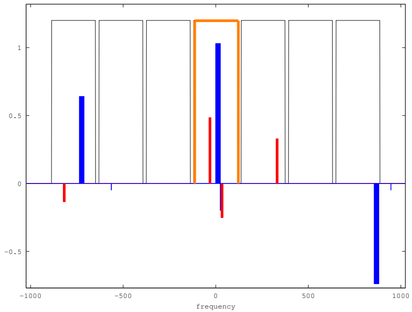
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

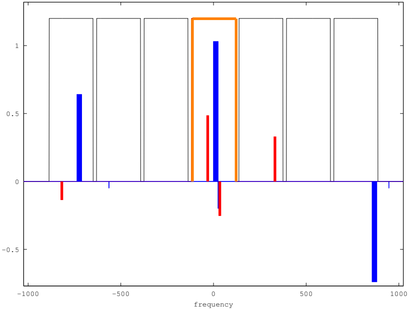
Zero-th bucket signal u^0:

$$\widehat{u}^0_f = \begin{cases} \widehat{x}_f, & \text{if } f \in [-\frac{n}{2B} : \frac{n}{2B}] \\ 0 & \text{o.w.} \end{cases}$$



Zero-th bucket signal u^0:

$$\widehat{u}^0_f = \begin{cases} \widehat{x}_f, & \text{if } f \in \left[-\frac{n}{2B} : \frac{n}{2B}\right] \\ 0 & \text{o.w.} \end{cases}$$

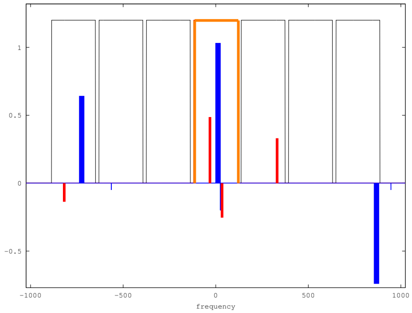


We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_f \widehat{u}^0_f \cdot \omega^{f \cdot a}$$

Zero-th bucket signal u^0:

$$\widehat{u}^0_f = \begin{cases} \widehat{x}_f, & \text{if } f \in \left[-\frac{n}{2B} : \frac{n}{2B}\right] \\ 0 & \text{o.w.} \end{cases}$$

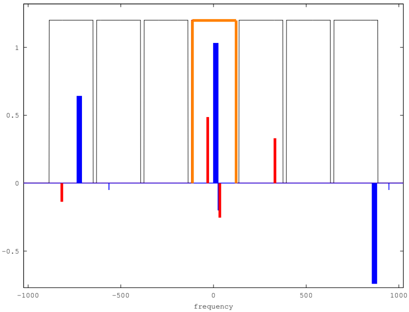


We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_f \widehat{u}^0_f \cdot \omega^{f \cdot a} = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{x}_f \cdot \omega^{f \cdot a}$$

Zero-th bucket signal u^0:

$$\widehat{u}^0_f = \begin{cases} \widehat{x}_f, & \text{if } f \in \left[-\frac{n}{2B} : \frac{n}{2B}\right] \\ 0 & \text{o.w.} \end{cases}$$



We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_f \widehat{u}^0_f \cdot \omega^{f \cdot a} = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{x}_f \cdot \omega^{f \cdot a} = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x by the time shift theorem).

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u_a^0 = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x).

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x).

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in [-\frac{n}{2B} : \frac{n}{2B}] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f$$

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x).

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in \left[-\frac{n}{2B} : \frac{n}{2B}\right] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \widehat{y}_f = \sum_{f \in [n]} \widehat{y}_f \widehat{G}_f$$

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x).

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in [-\frac{n}{2B} : \frac{n}{2B}] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f = \sum_{f \in [n]} \widehat{y}_f \widehat{G}_f = (\widehat{y} * \widehat{G})(0)$$

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f,$$

where $y_j = x_{j+a}$ (y is a time shift of x).

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in [-\frac{n}{2B} : \frac{n}{2B}] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{y}_f = \sum_{f \in [n]} \widehat{y}_f \widehat{G}_f = (\widehat{y} * \widehat{G})(0) = (\widehat{x_{\cdot+a}} * \widehat{G})(0)$$

Need to evaluate

$$(\widehat{x} * \widehat{G})\left(\mathbf{j} \cdot \frac{\mathbf{n}}{\mathbf{B}}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

Need to evaluate

$$(\widehat{x} * \widehat{G})\left(\mathbf{j} \cdot \frac{\mathbf{n}}{\mathbf{B}}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

By the convolution identity

$$\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$$

Need to evaluate

$$(\widehat{x} * \widehat{G})\left(\mathbf{j} \cdot \frac{\mathbf{n}}{\mathbf{B}}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

By the convolution identity

$$\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$$

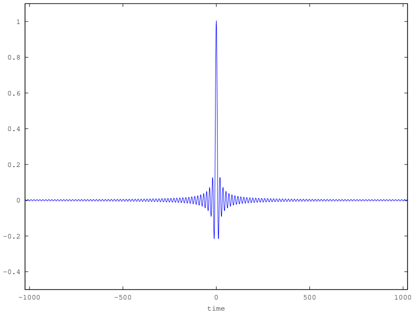
Suffices to compute

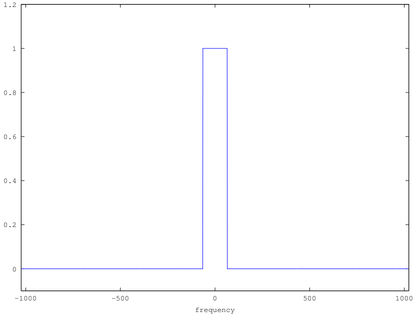
$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$

Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = -B/2, \ldots, B/2-1$$

Sample complexity? Runtime?

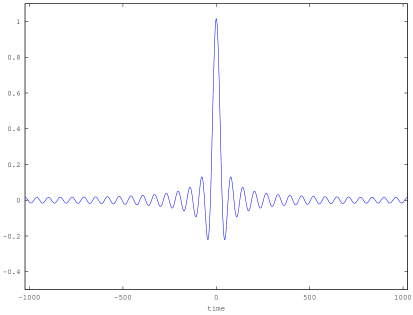


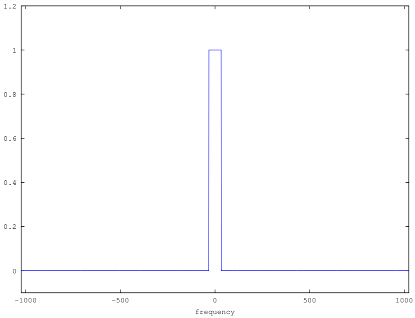


Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = -B/2, \ldots, B/2 - 1$$

Sample complexity? Runtime?

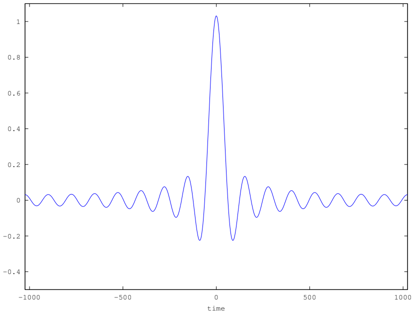


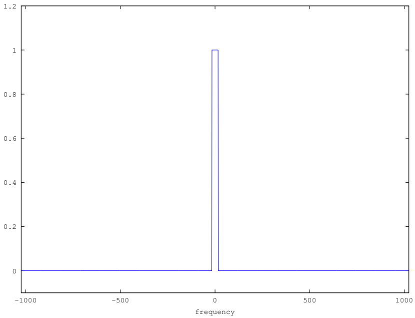


Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = -B/2, \ldots, B/2 - 1$$

Sample complexity? Runtime?

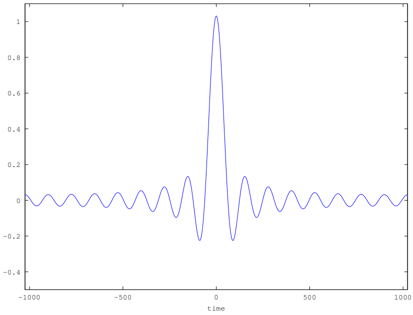


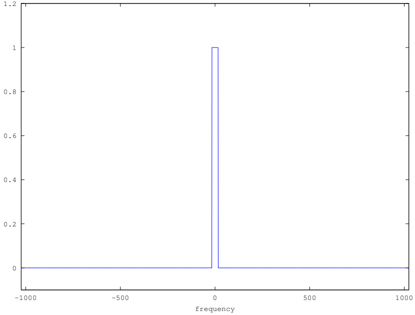


Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = -B/2, \ldots, B/2 - 1$$

Sample complexity? Runtime?



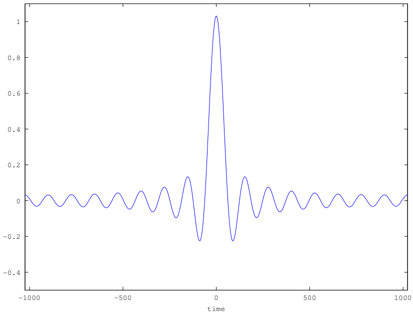


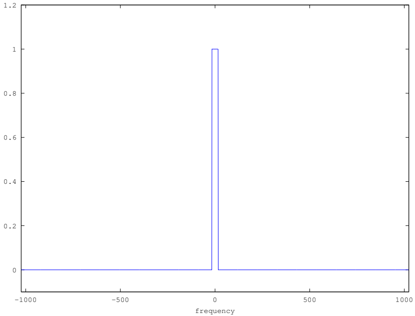
Computing $x \cdot G$ takes $\Omega(N)$ time and samples!

Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = -B/2, \ldots, B/2 - 1$$

Sample complexity? Runtime?





Computing $x \cdot G$ takes $\Omega(N)$ time and samples!

Design a filter $\mathrm{supp}(G) \approx k$? Truncate sinc? Tolerate imprecise hashing? Collisions in buckets?