Sparse Fourier Transform (lecture 2)

Michael Kapralov[1]

[1] IBM Watson

MADALGO'15

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

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

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 of x:

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

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

Goal: find the top k coefficients of $\widehat{x}$ approximately

In last lecture:

- 1-sparse noiseless case: two-point sampling

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

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

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

Goal: find the top k coefficients of $\widehat{x}$ approximately

In last lecture:

- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: $O(\log n \log\log n)$ time and samples

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

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

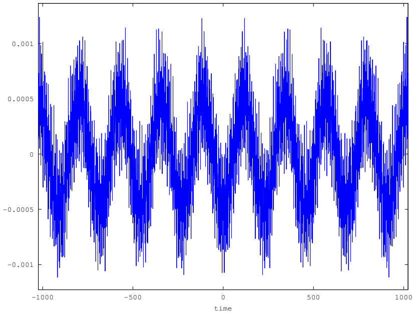
where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

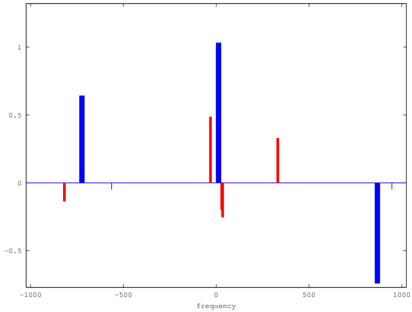
Goal: find the top k coefficients of $\widehat{x}$ approximately

In last lecture:

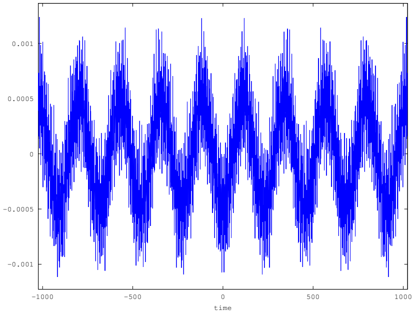
- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: $O(\log n \log\log n)$ time and samples
- reduction from k-sparse to 1-sparse case, via filtering

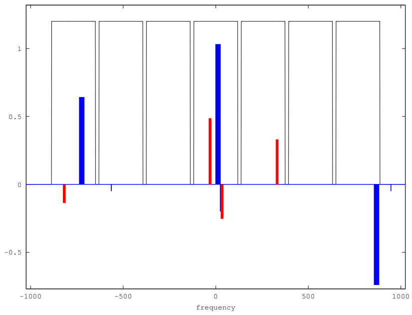
Partition frequency domain into $B \approx k$ buckets



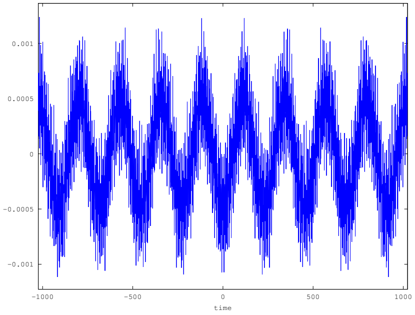


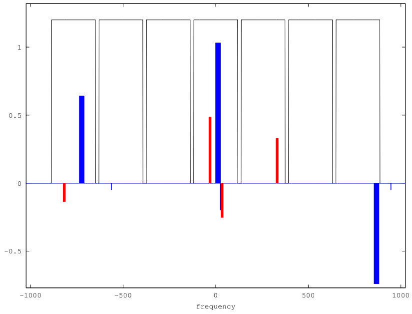
Partition frequency domain into $B \approx k$ buckets





Partition frequency domain into $B \approx k$ buckets



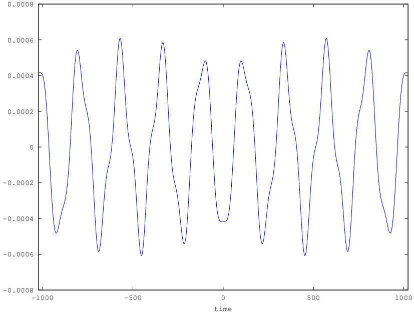


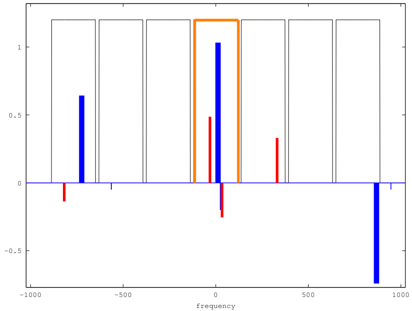
For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets



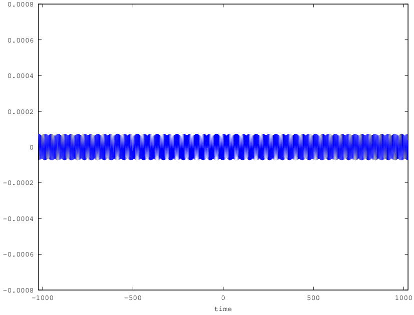


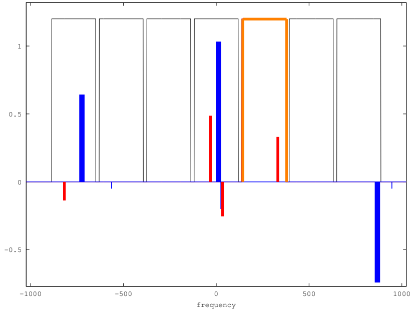
For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets



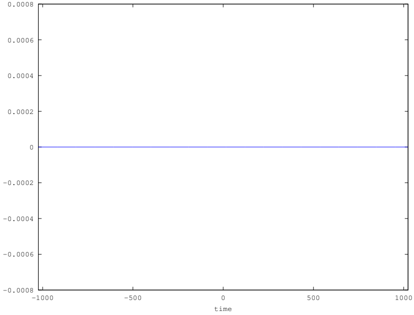


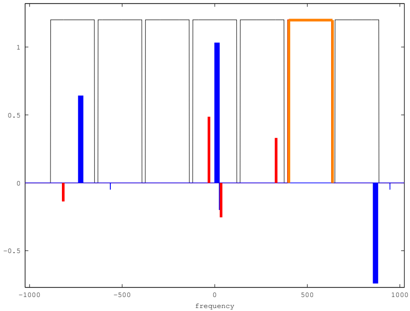
For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets



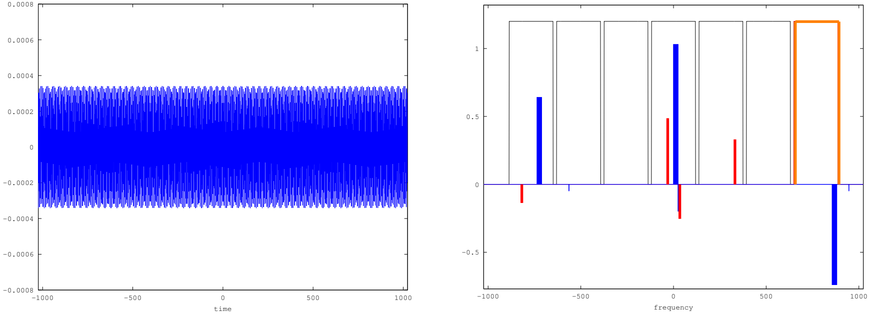


For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets

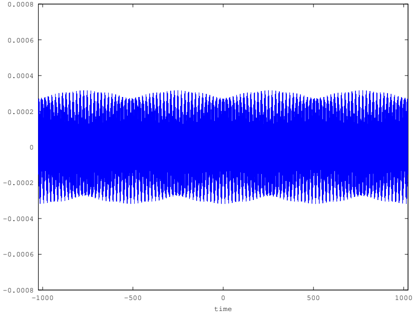


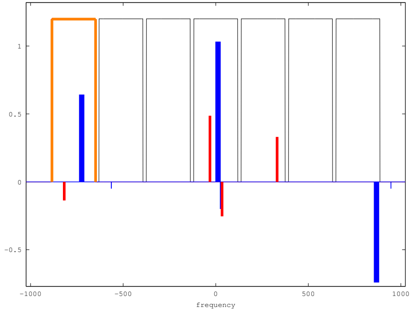
For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets



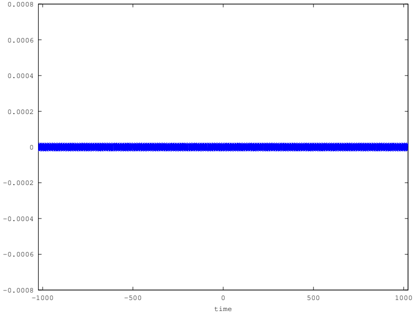


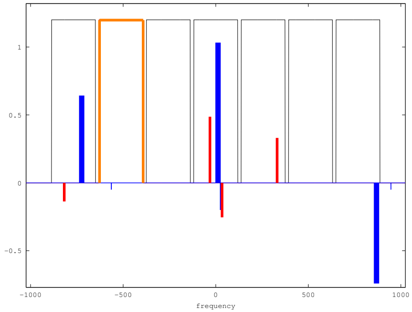
For each $j = 0, \ldots, B-1$ let

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

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

Partition frequency domain into $B \approx k$ buckets



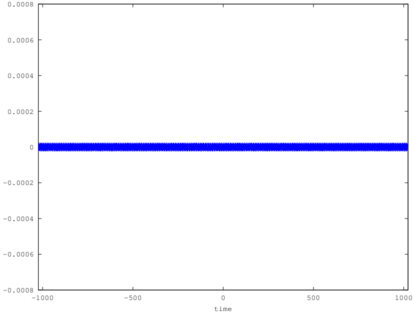


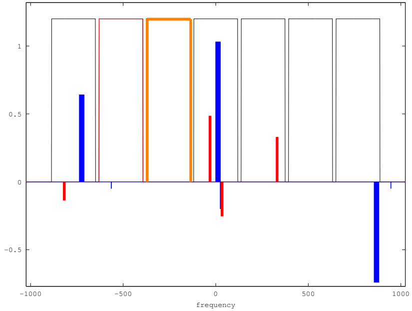
For each $j = 0, \ldots, B-1$ let

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

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!

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

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

Let

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

Then

$$u^0_a = (\widehat{x_{\cdot + a}} * \widehat{G})(0)$$

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

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

Let

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

Then

$$u^0_a = (\widehat{x_{\cdot + a}} * \widehat{G})(0)$$

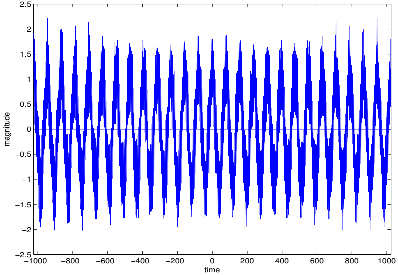
For any $j = 0, \ldots, B-1$

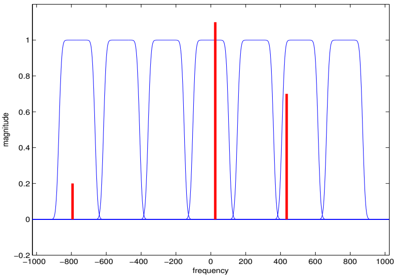
$$u^j_a = (\widehat{x_{\cdot + a}} * \widehat{G})(j \cdot \frac{n}{B})$$

Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_a^j = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$

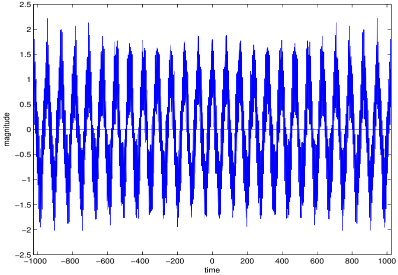


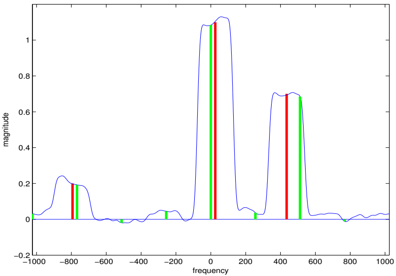


Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_a^j = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$

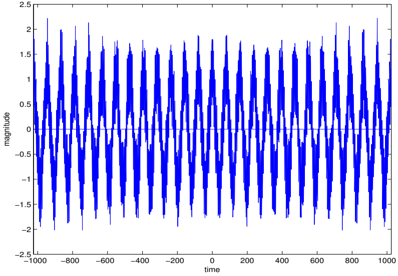


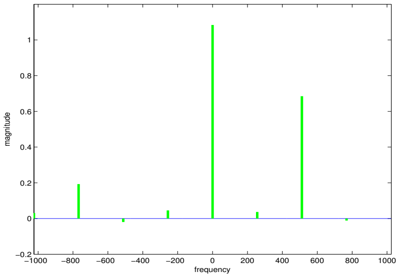


Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_{a}^{j} = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$





Need to evaluate

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

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

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

Need to evaluate

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

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

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

By the convolution identity

$$\widehat{x}_{\cdot+a} * \widehat{G} = \left(\widehat{x_{\cdot+a} \cdot G}\right)$$

Need to evaluate

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

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

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

By the convolution identity

$$\widehat{x}_{\cdot+a} * \widehat{G} = \left(\widehat{x_{\cdot+a} \cdot G}\right)$$

Suffices to compute

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

Suffices to compute

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

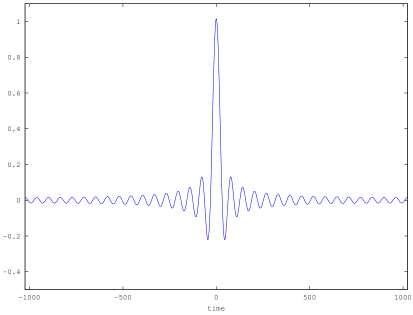
Suffices to compute

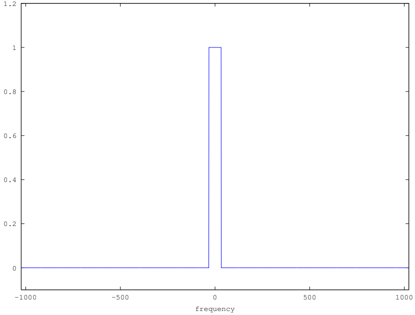
$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B - 1$$

Suffices to compute

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

Sample complexity? Runtime?

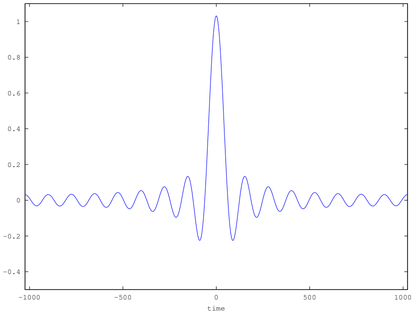


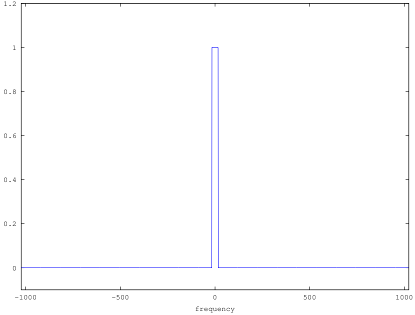


Suffices to compute

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

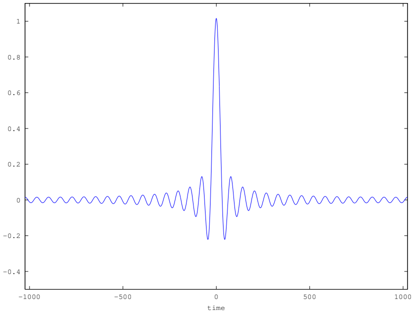
Sample complexity? Runtime?

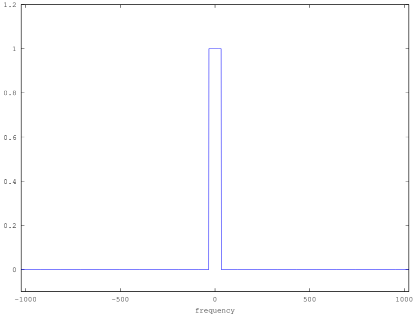




To sample all signals $u^j, j = 0, \ldots, B-1$ in time domain, it suffices to compute

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





Computing $x \cdot G$ takes $\mathrm{supp}(G)$ samples.

Design G with $\mathrm{supp}(G) \approx k$ that approximates rectangular filter?

In this lecture:

- permuting frequencies
- filter construction
- recovery algorithm (k-sparse noiseless case)

1. Pseudorandom spectrum permutations
2. Filter construction
3. Basic block: partial recovery
4. Full algorithm

1. **Pseudorandom spectrum permutations**
2. Filter construction
3. Basic block: partial recovery
4. Full algorithm

Pseudorandom spectrum permutations

Permutation in time domain plus phase shift $\Longrightarrow$ permutation in frequency domain

Pseudorandom spectrum permutations

Permutation in time domain plus phase shift $\Longrightarrow$ permutation in frequency domain

Claim

Let $\sigma, b \in [n]$, σ *invertible modulo n. Let* $y_j = x_{\sigma j}\omega^{-jb}$. *Then*

$$\widehat{y}_f = \widehat{x}_{\sigma^{-1}(f+b)}.$$

(proof on next slide; a close relative of time shift theorem)

Pseudorandom spectrum permutations

Permutation in time domain plus phase shift $\Longrightarrow$ permutation in frequency domain

Claim

Let $\sigma, b \in [n]$, σ invertible modulo n. Let $y_j = x_{\sigma j}\omega^{-jb}$. Then

$$\widehat{y}_f = \widehat{x}_{\sigma^{-1}(f+b)}.$$

(proof on next slide; a close relative of time shift theorem)

Pseudorandom permutation:

- select b uniformly at random from $[n]$
- select σ uniformly at random from $\{1, 3, 5, \ldots, n-1\}$ (invertible numbers modulo n)

Pseudorandom spectrum permutations

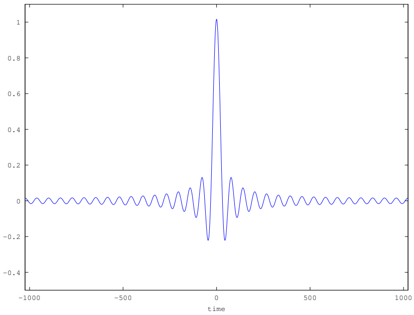
Claim

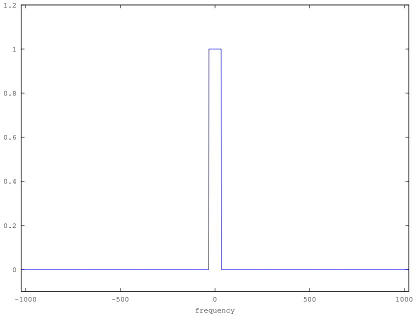
Let $y_j = x_{\sigma j}\omega^{-jb}$. Then $\widehat{y}_f = \widehat{x}_{\sigma^{-1}(f+b)}$.

Proof.

$$\begin{aligned}
\widehat{y}_f &= \frac{1}{n}\sum_{j\in[n]} y_j \omega^{-f\cdot j} \\
&= \frac{1}{n}\sum_{j\in[n]} x_{\sigma j}\omega^{-(f+b)\cdot j} \\
&= \frac{1}{n}\sum_{i\in[n]} x_i \omega^{-(f+b)\cdot\sigma^{-1} i} \quad \text{(change of variables } i = \sigma j\text{)} \\
&= \frac{1}{n}\sum_{i\in[n]} x_i \omega^{-\sigma^{-1}(f+b)\cdot i} \\
&= \widehat{x}_{\sigma^{-1}(f+b)}
\end{aligned}$$

$\square$

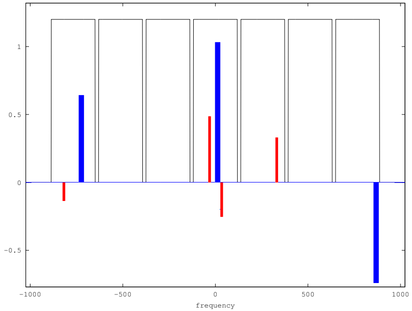




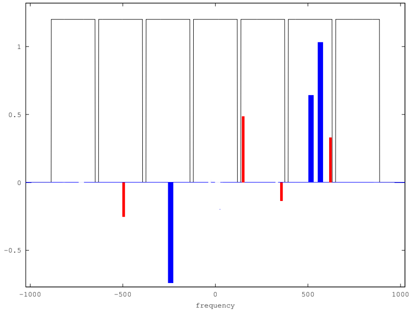
Design G with $\text{supp}(G) \approx k$ that approximates rectangular filter?

Our filter $\widehat{G}$ will approximate the boxcar. Bound collision probability now.

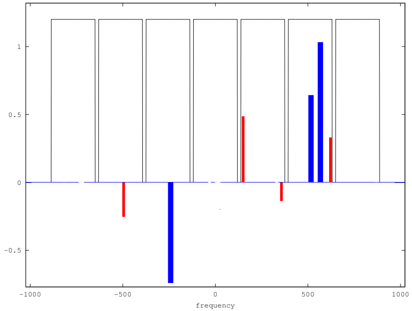
Partition frequency domain into buckets, permute spectrum



Partition frequency domain into buckets, permute spectrum

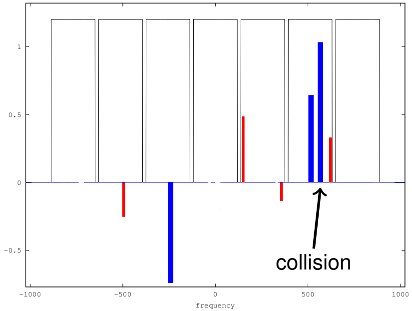


Partition frequency domain into buckets, permute spectrum



Frequency i collides with frequency j only if $|\sigma i - \sigma j| \leq \frac{n}{B}$.

Partition frequency domain into buckets, permute spectrum



Frequency i collides with frequency j only if $|\sigma i - \sigma j| \leq \frac{n}{B}$.

Collision probability

Lemma

Let σ be a uniformly random odd number in $1, 2, \ldots, n$. Then for any $i, j \in [n], i \neq j$ one has

$$\mathbf{Pr}_{\sigma}\left[|\sigma \cdot i - \sigma j| \leq \frac{n}{B}\right] = O(1/B)$$

Collision probability

Lemma

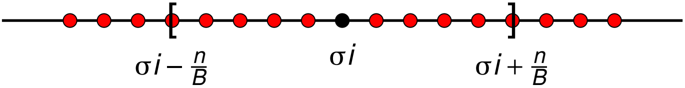
Let σ be a uniformly random odd number in $1,2,\ldots,n$. Then for any $i,j \in [n], i \neq j$ one has

$$\mathbf{Pr}_{\sigma}\left[|\sigma \cdot i - \sigma j| \leq \frac{n}{B}\right] = O(1/B)$$

Proof.

Let $\Delta := i - j = d2^s$ for some odd d.

The orbit of $\sigma \cdot \Delta$ is $2^s \cdot d'$ for all odd d'.



There are $O(\frac{n}{B2^s})$ values of d' that make $\sigma \cdot \Delta$ fall into $[-\frac{n}{B}, \frac{n}{B}]$, out of $n/2^{s+1}$. □

Collision probability

Lemma

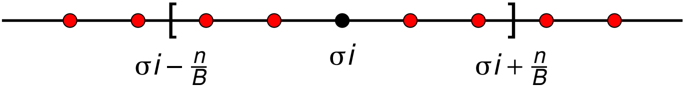
Let σ be a uniformly random odd number in $1, 2, \ldots, n$. Then for any $i, j \in [n], i \neq j$ one has

$$\mathbf{Pr}_{\sigma}\left[|\sigma \cdot i - \sigma j| \leq \frac{n}{B}\right] = O(1/B)$$

Proof.

Let $\Delta := i - j = d2^s$ for some odd d.

The orbit of $\sigma \cdot \Delta$ is $2^s \cdot d'$ for all odd d'.



There are $O(\frac{n}{B2^s})$ values of d' that make $\sigma \cdot \Delta$ fall into $[-\frac{n}{B}, \frac{n}{B}]$, out of $n/2^{s+1}$. $\square$

Collision probability

Lemma

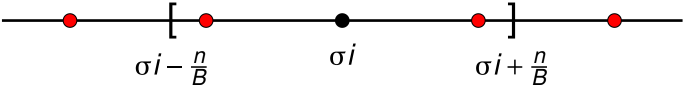
Let σ be a uniformly random odd number in $1, 2, \ldots, n$*. Then for any* $i, j \in [n], i \neq j$ *one has*

$$\mathbf{Pr}_\sigma \left[|\sigma \cdot i - \sigma j| \leq \frac{n}{B} \right] = O(1/B)$$

Proof.

Let $\Delta := i - j = d2^s$ for some odd d.

The orbit of $\sigma \cdot \Delta$ is $2^s \cdot d'$ for all odd d'.



There are $O(\frac{n}{B2^s})$ values of d' that make $\sigma \cdot \Delta$ fall into $[-\frac{n}{B}, \frac{n}{B}]$, out of $n/2^{s+1}$. $\square$

Collision probability

Lemma

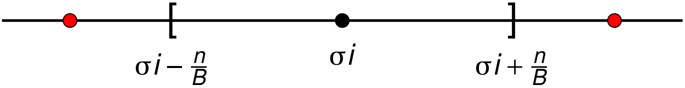
Let σ be a uniformly random odd number in $1,2,\ldots,n$. Then for any $i,j \in [n], i \neq j$ one has

$$\mathbf{Pr}_\sigma\left[|\sigma \cdot i - \sigma j| \le \frac{n}{B}\right] = O(1/B)$$

Proof.

Let $\Delta := i - j = d2^s$ for some odd d.

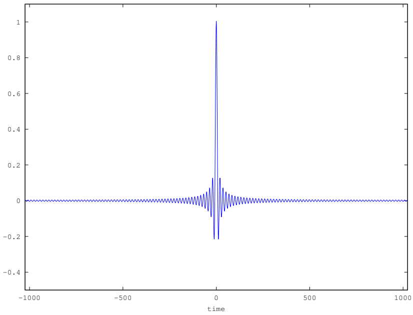
The orbit of $\sigma \cdot \Delta$ is $2^s \cdot d'$ for all odd d'.

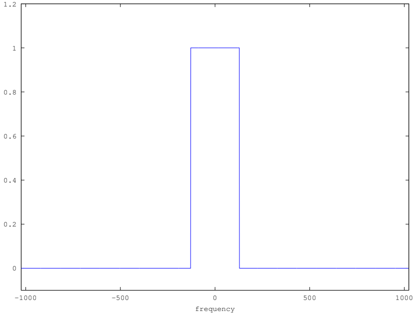


There are $O(\frac{n}{B2^s})$ values of d' that make $\sigma \cdot \Delta$ fall into $[-\frac{n}{B}, \frac{n}{B}]$, out of $n/2^{s+1}$. $\square$

1. Pseudorandom spectrum permutations
2. **Filter construction**
3. Basic block: partial recovery
4. Full algorithm

Rectangular buckets $\widehat{G}$ have full support in time domain...



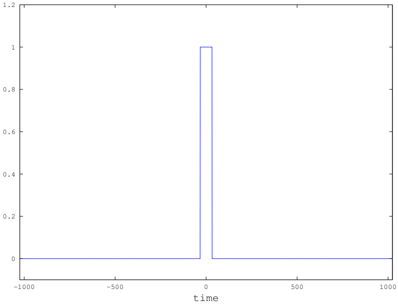


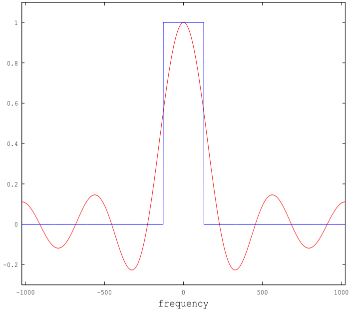
$\boxed{\text{Approximate rectangular filter with a filter } G \text{ with small support?}}$

Need $\mathrm{supp}(G) \approx k$, so perhaps turn the filter around?

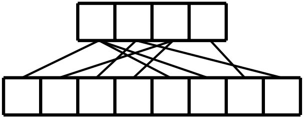
Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Have $\mathrm{supp}(G) = B \approx k$, but buckets leak



$\Longrightarrow$

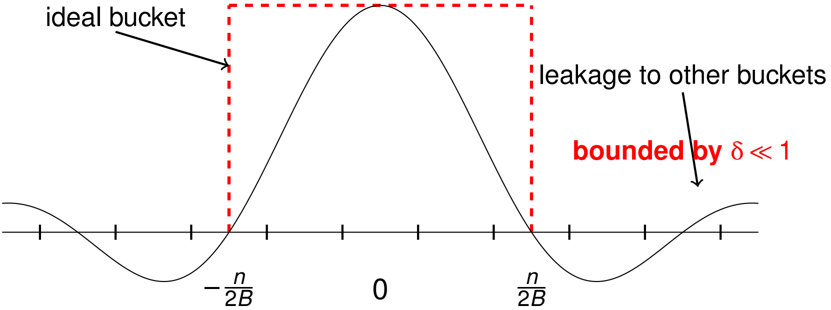
In what follows: reduce leakage at the expense of increasing supp(G)

Window functions

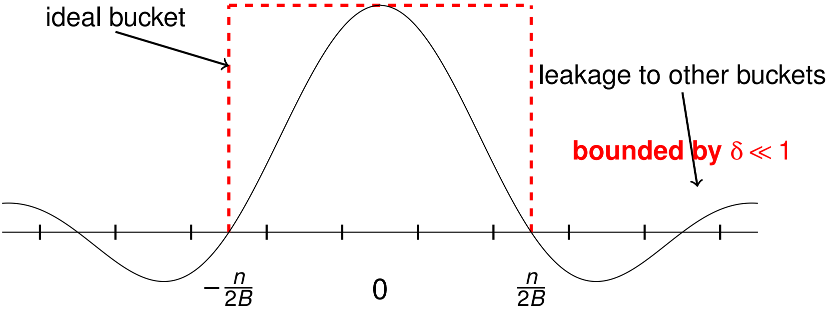
Definition

A symmetric filter G is a (B, δ)-*standard window function* if

1. $\widehat{G}_0 = 1$
2. $\widehat{G}_f \geq 0$
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$



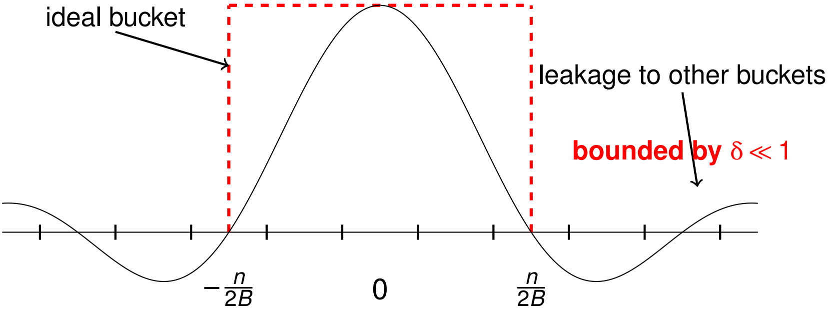
Window functions



Start with the sinc function:

$$\widehat{G}_f := \frac{\sin(\pi(B+1)f/n)}{(B+1)\cdot\pi f/n}$$

Window functions



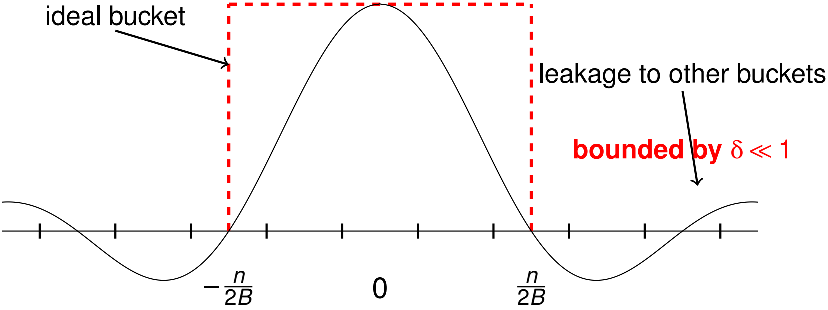
Start with the sinc function:

$$\widehat{G}_f := \frac{\sin(\pi(B+1)f/n)}{(B+1)\cdot\pi f/n}$$

For all $|f| > \frac{n}{2B}$ we have

$$|\widehat{G}_f| \leq \frac{1}{(B+1)\pi f/n} \leq \frac{1}{\pi/2} \leq 2/\pi \leq 0.9$$

Window functions



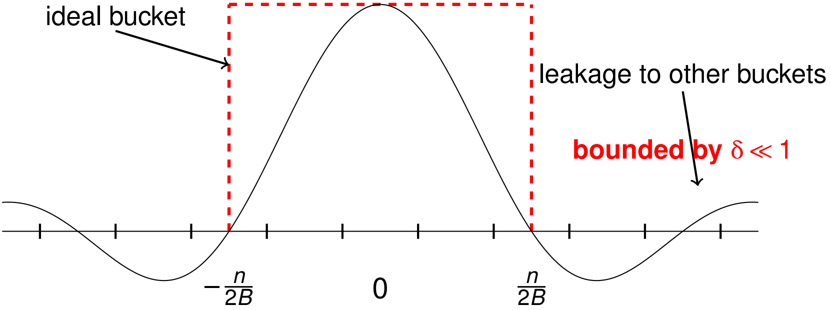
Consider **powers of the sinc function**:

$$\widehat{G}_f^r := \left(\frac{\sin(\pi(B+1)f/n)}{(B+1)\cdot\pi f/n}\right)^r$$

For all $|f| > \frac{n}{2B}$ we have

$$|\widehat{G}_f|^r \le (0.9)^r$$

Window functions



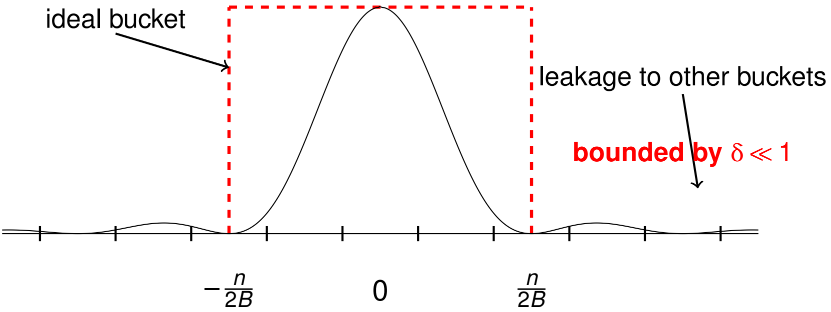
Consider **powers of the sinc function**:

$$\widehat{G}_f^r := \left(\frac{\sin(\pi(B+1)f/n)}{(B+1)\cdot\pi f/n}\right)^r$$

For all $|f| > \frac{n}{2B}$ we have

$$|\widehat{G}_f|^r \leq (0.9)^r$$

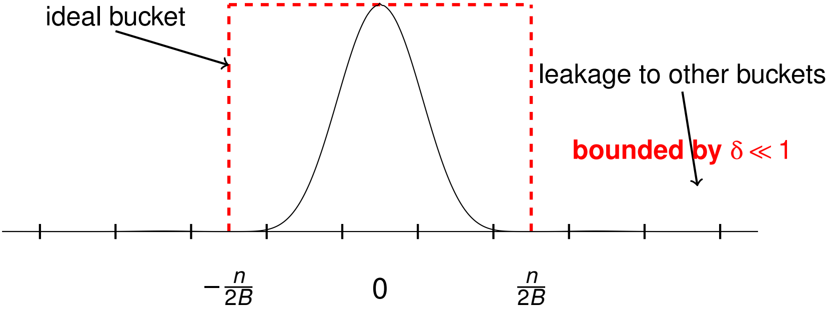
Window functions



Consider **powers of the sinc function**: $\widehat{G}_f^r$
For all $|f| > \frac{n}{2B}$ we have

$$|\widehat{G}_f|^r \leq (0.9)^r$$

Window functions



Consider **powers of the sinc function**: $\widehat{G}_f^r$
For all $|f| > \frac{n}{2B}$ we have

$$|\widehat{G}_f|^r \leq (0.9)^r$$

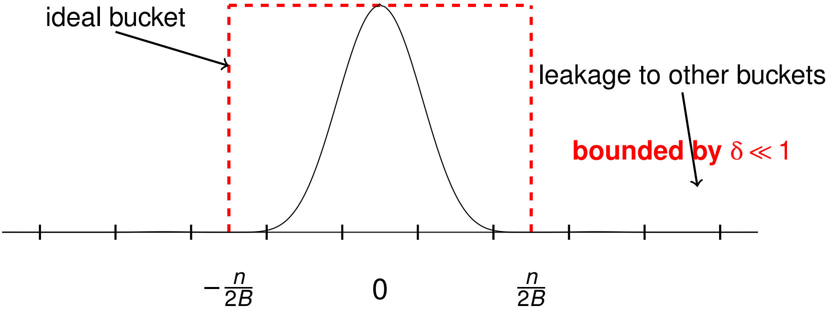
$\boxed{\textbf{So setting } r = O(\log(1/\delta)) \textbf{ is sufficient!}}$

Window functions

Definition

A symmetric filter G is a (B, δ)-*standard window function* if

1. $\widehat{G}_0 = 1$
2. $\widehat{G}_f \geq 0$
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$

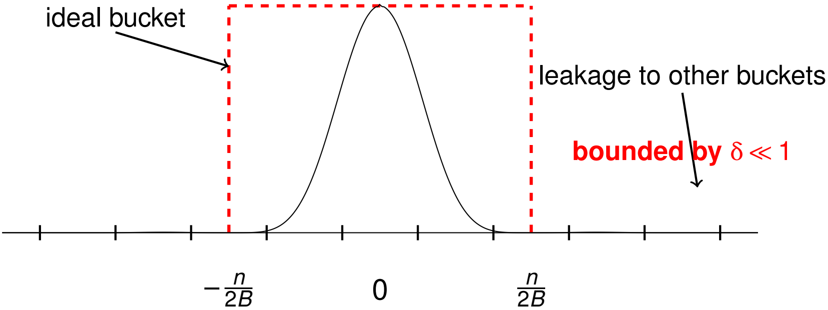


Window functions

Definition

A symmetric filter G is a (B, δ)-*standard window function* if

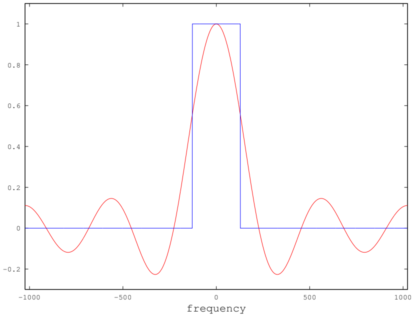
1. $\widehat{G}_0 = 1$
2. $\widehat{G}_f \geq 0$
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$

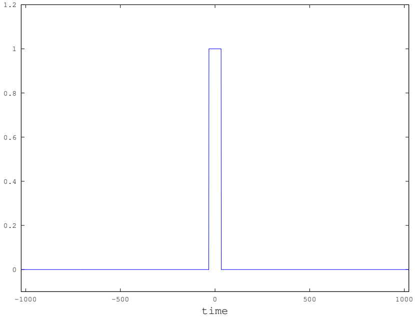


How large is supp$(G) \subseteq [-T, T]$**?**

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$

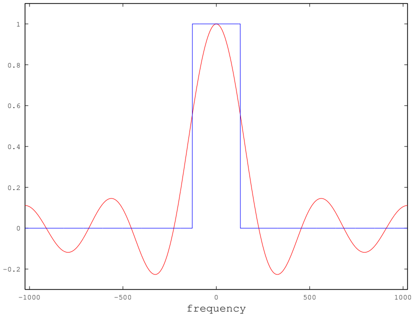


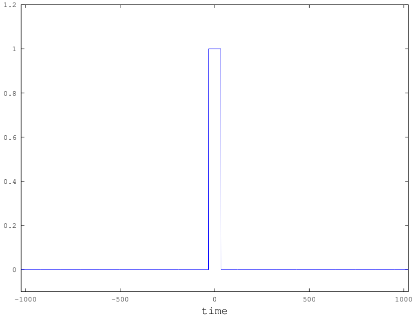


Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$



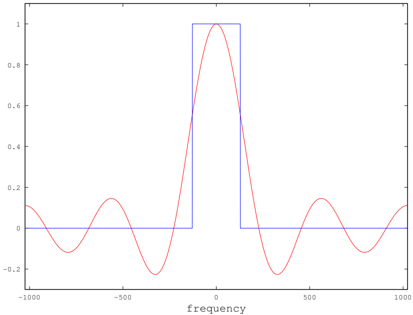


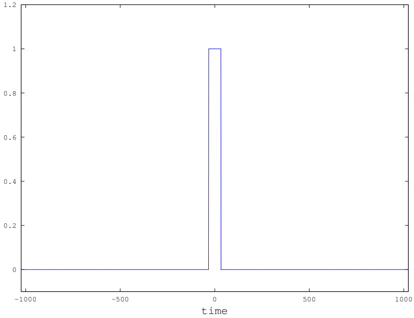
Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

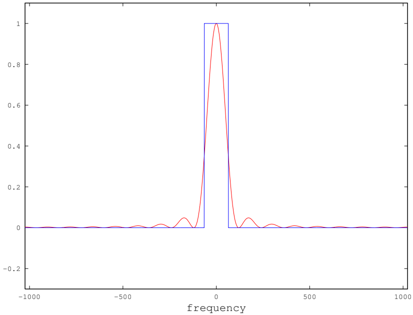
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

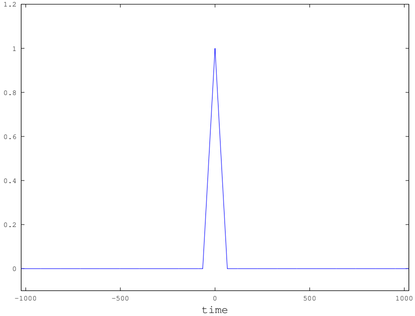
Support of G^0 is in $[-B/2, B/2]$, so

$$\operatorname{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

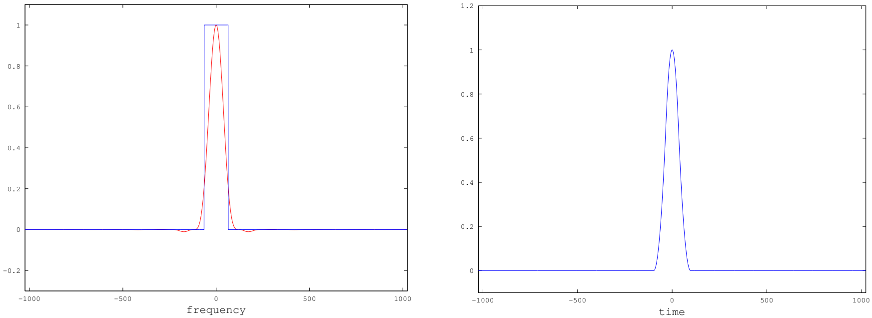
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

Support of G^0 is in $[-B/2, B/2]$, so

$$\operatorname{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$



Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

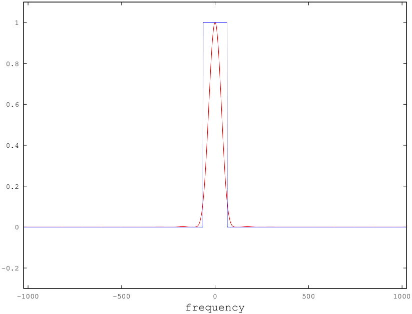
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

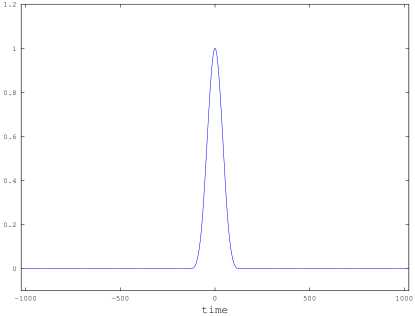
Support of G^0 is in $[-B/2, B/2]$, so

$$\mathrm{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

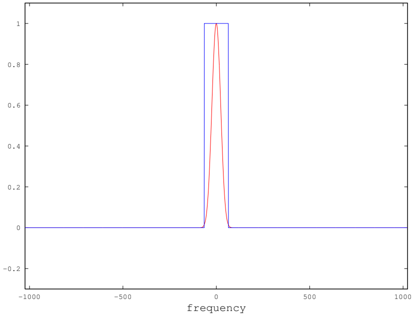
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

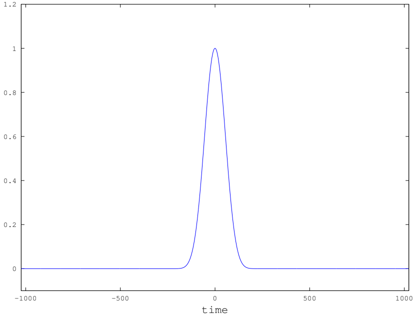
Support of G^0 is in $[-B/2, B/2]$, so

$$\text{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

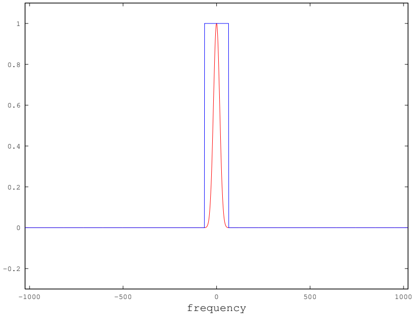
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

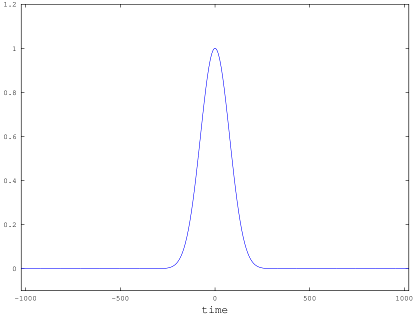
Support of G^0 is in $[-B/2, B/2]$, so

$$\text{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

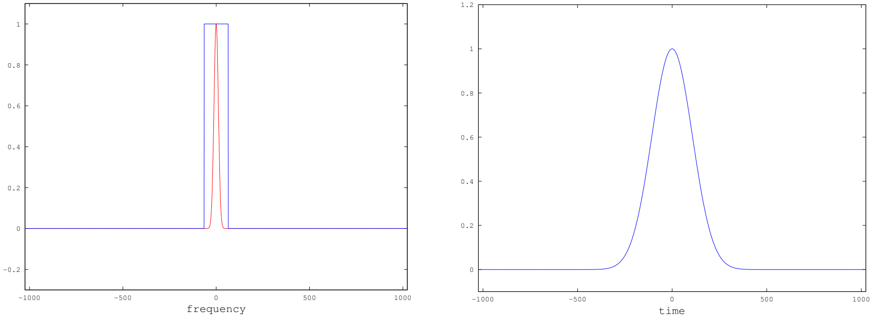
By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

Support of G^0 is in $[-B/2, B/2]$, so

$$\text{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$



Let $\widehat{G}^r := (\widehat{G}^0)^r$. How large is the support of G^r?

By the convolution identity $G^r = G^0 * G^0 * \ldots * G^0$

Support of G^0 is in $[-B/2, B/2]$, so

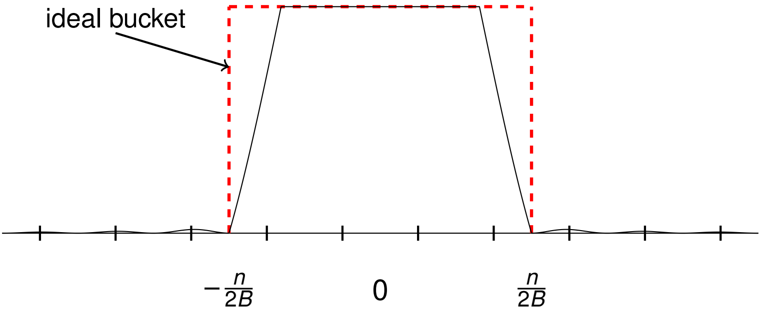
$$\text{supp}(G * \ldots * G) \subseteq [-r \cdot B/2, r \cdot B/2]$$

Flat window function

Definition

A symmetric filter G is a (B, δ, γ)-**flat** *window function* if

1. $\widehat{G}_j \geq 1 - \delta$ for all $j \in [-(\mathbf{1}-\gamma)\frac{n}{2B}, (\mathbf{1}-\gamma)\frac{n}{2B}]$
2. $\widehat{G}_j \in [0, 1]$ for all j
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$

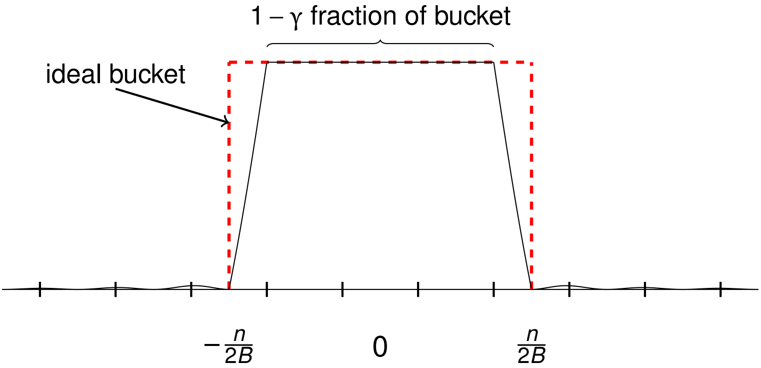


Flat window function

Definition

A symmetric filter G is a (B, δ, γ)-**flat** *window function* if

1. $\widehat{G}_j \geq 1 - \delta$ for all $j \in [-(\mathbf{1} - \gamma)\frac{n}{2B}, (\mathbf{1} - \gamma)\frac{n}{2B}]$
2. $\widehat{G}_j \in [0, 1]$ for all j
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$

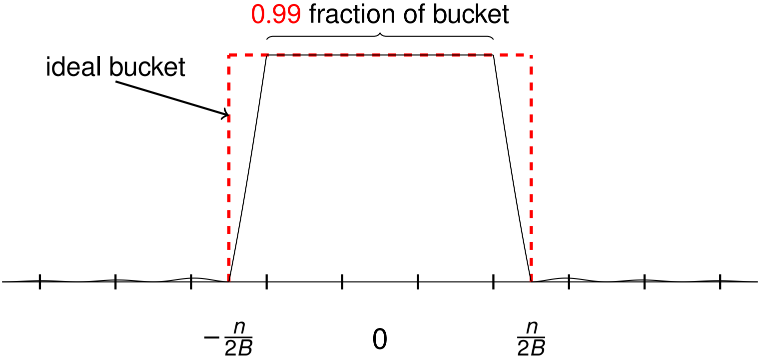


Flat window function

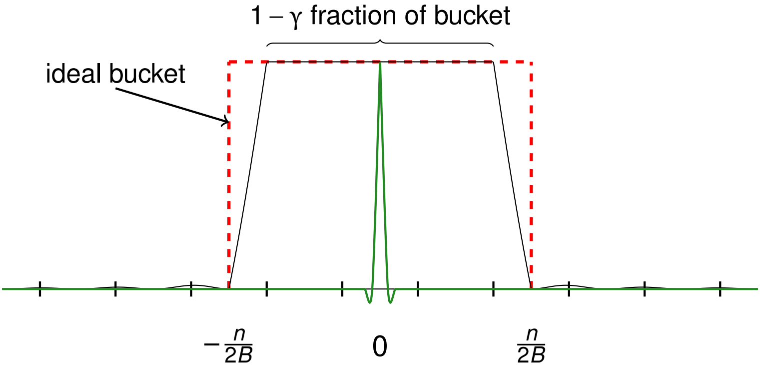
Definition

A symmetric filter G is a (B, δ, γ)-**flat** *window function* if

1. $\widehat{G}_j \geq 1 - \delta$ for all $j \in [-(\mathbf{1} - \gamma)\frac{n}{2B}, (\mathbf{1} - \gamma)\frac{n}{2B}]$
2. $\widehat{G}_j \in [0, 1]$ for all j
3. $|\widehat{G}_f| \leq \delta$ for $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$



Flat window function – construction



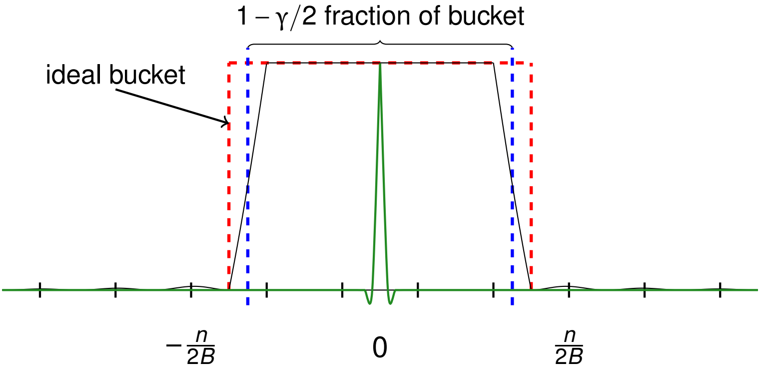
Let H be a $(2B/\gamma, \delta/n)$-standard window function. Note that

$$|\widehat{H}_f| \le \delta/n$$

for all f outside of

$$\left[-\gamma\frac{n}{4B}, \gamma\frac{n}{4B}\right].$$

Flat window function – construction



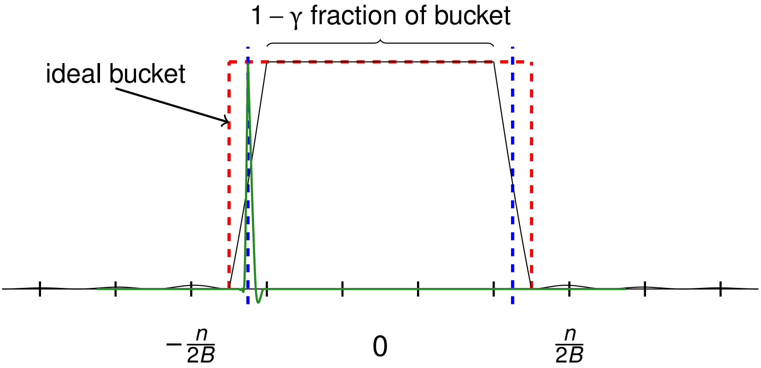
Let H be a $(2B/\gamma, \delta/n)$-standard window function. Note that

$$|\widehat{H}_f| \leq \delta/n$$

for all f outside of

$$\left[-\gamma\frac{n}{4B}, \gamma\frac{n}{4B}\right].$$

Flat window function – construction



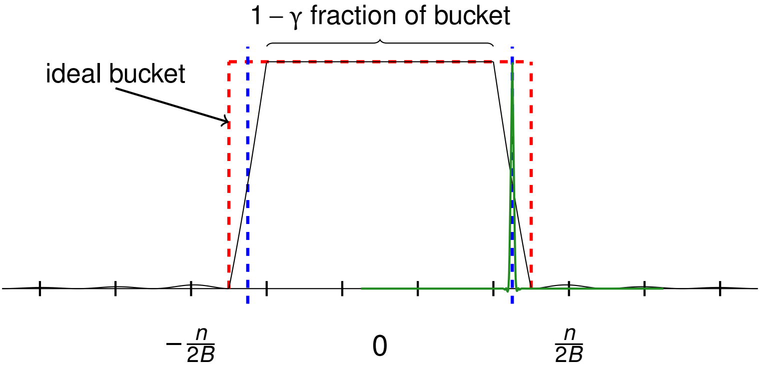
Let H be a $(2B/\gamma, \delta/n)$-standard window function. Note that

$$|\widehat{H}_f| \leq \delta/n$$

for all f outside of

$$\left[-\gamma\frac{n}{4B}, \gamma\frac{n}{4B}\right].$$

Flat window function – construction



Let H be a $(2B/\gamma, \delta/n)$-standard window function. Note that

$$|\widehat{H}_f| \leq \delta/n$$

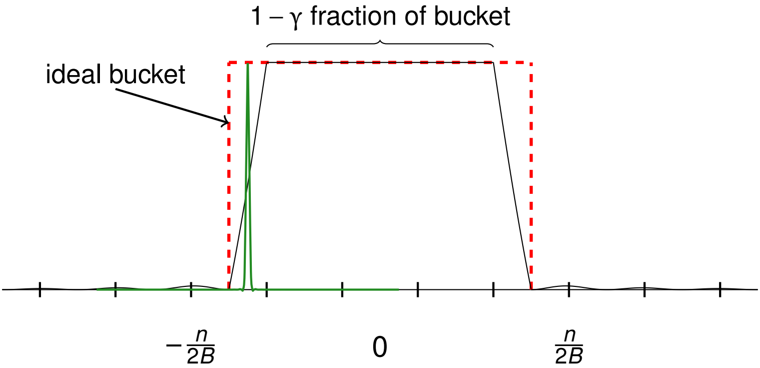
for all f outside of

$$\left[-\gamma\frac{n}{4B}, \gamma\frac{n}{4B}\right].$$

Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



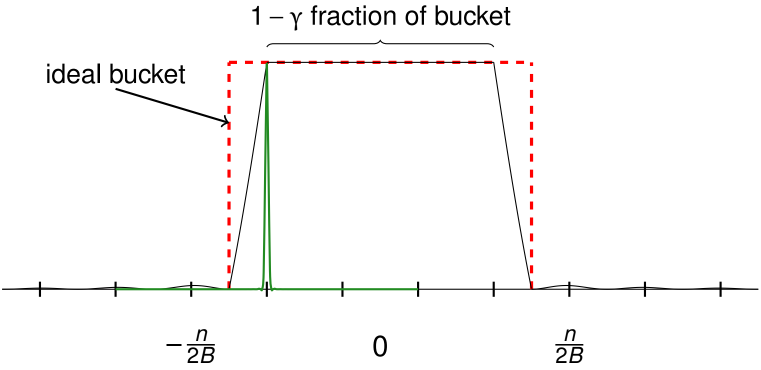
Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1 - \gamma/2)\frac{n}{2B}$$

2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



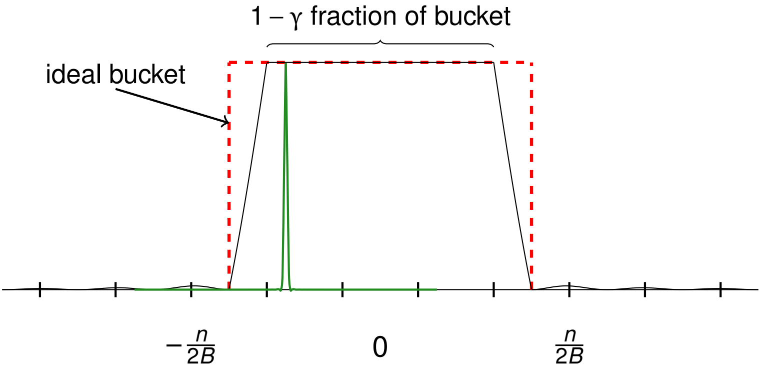
Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1-\gamma/2)\frac{n}{2B}$$

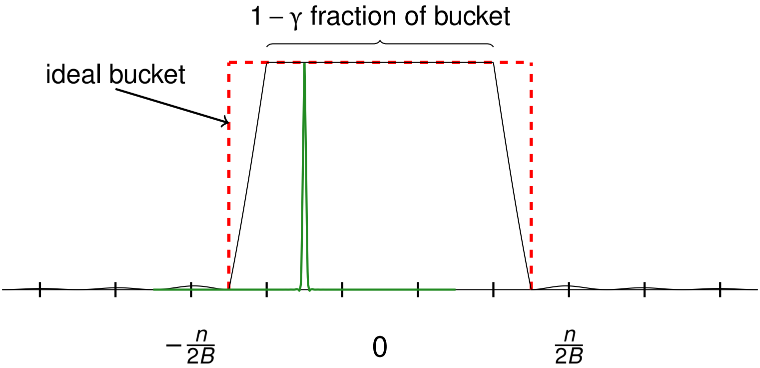
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

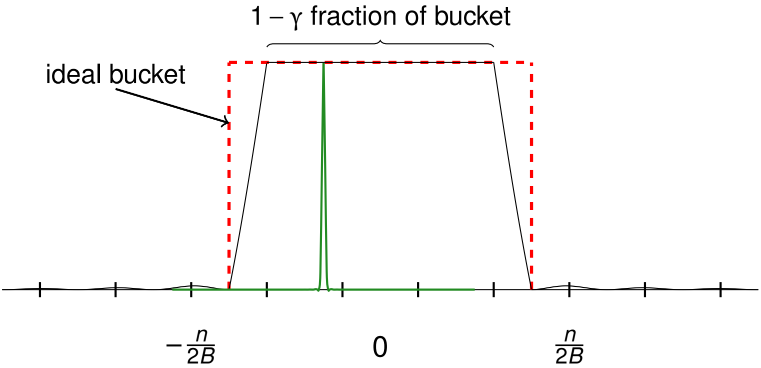
1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

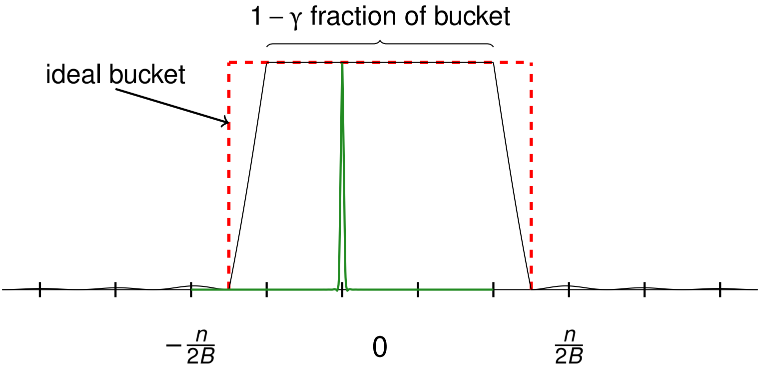
1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

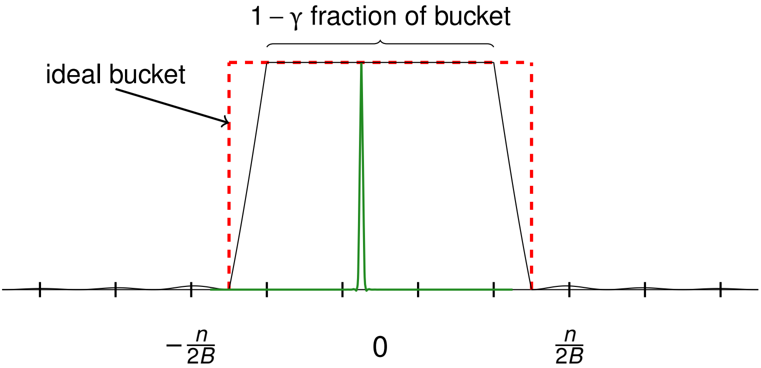
1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



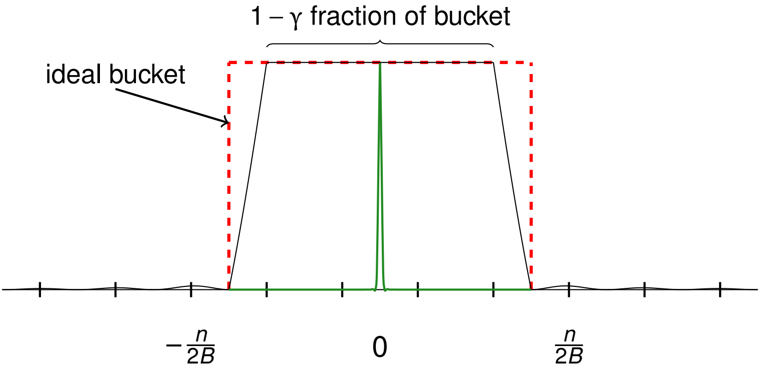
Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1 - \gamma/2)\frac{n}{2B}$$

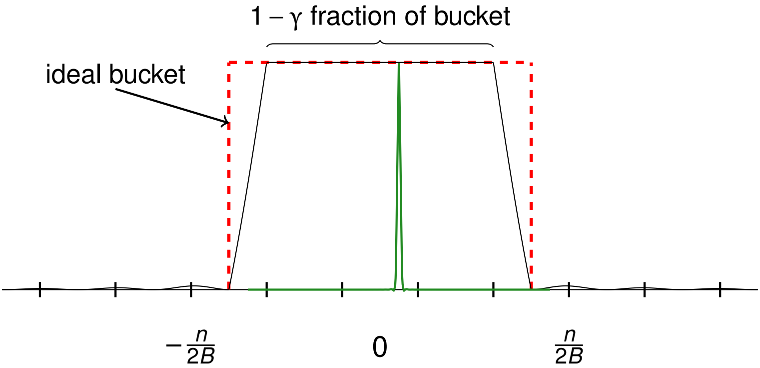
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



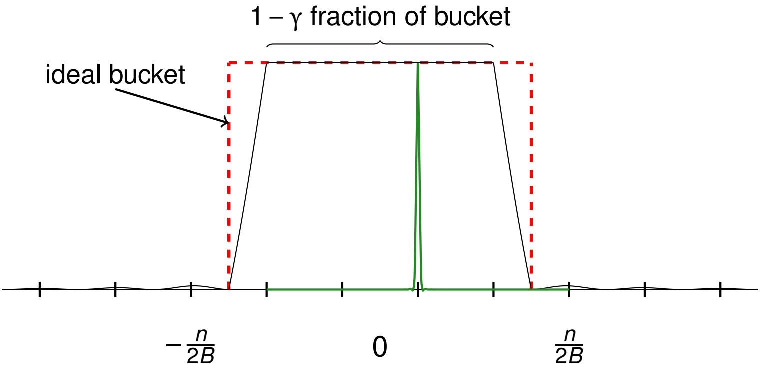
Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1 - \gamma/2)\frac{n}{2B}$$

2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



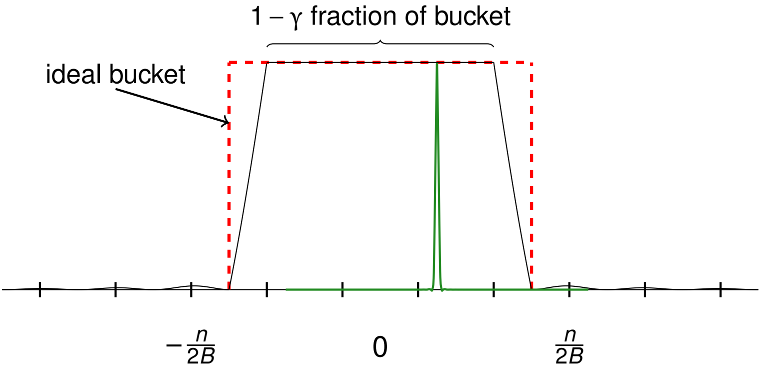
Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1-\gamma/2)\frac{n}{2B}$$

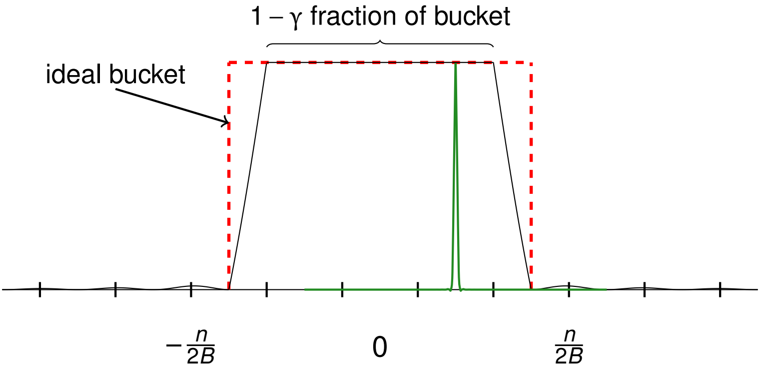
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

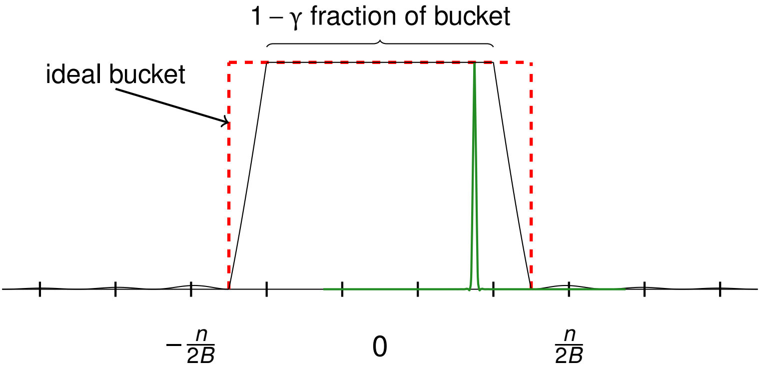
1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

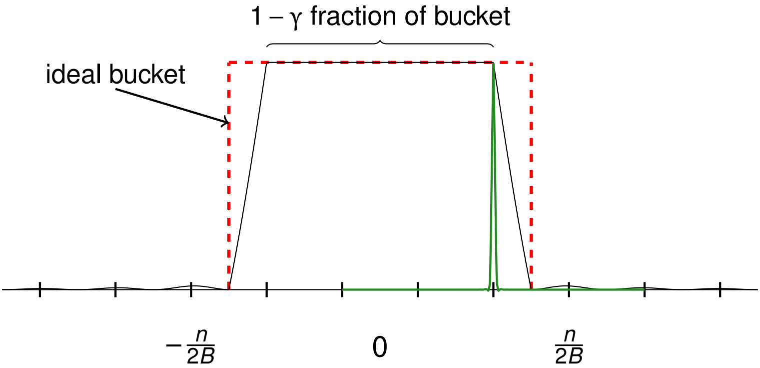
1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1-\gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

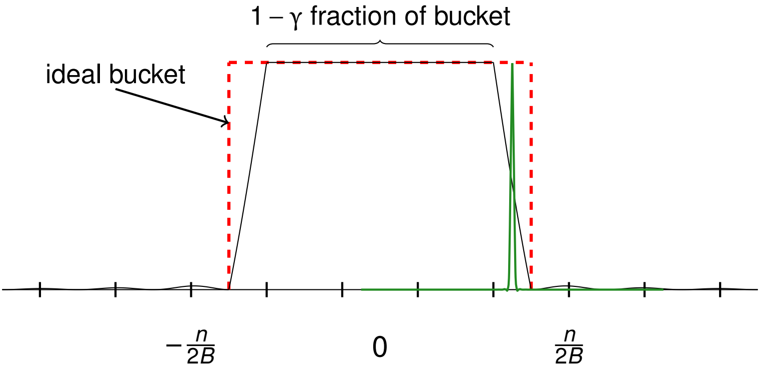
1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



Flat window function – construction

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot-\Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$



To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where

$$U = (1 - \gamma/2)\frac{n}{2B}$$

2. normalize so that $\widehat{G}_0 = 1 \pm \delta$

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$

Formally:
$$\widehat{G}_f := \frac{1}{Z}\left(\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U}\right)$$
where Z is a normalization factor.

To construct $\widehat{G}$:

1. sum up shifts $\widehat{H}_{\cdot - \Delta}$ over all $\Delta \in [-U, U]$, where
$$U = (1 - \gamma/2)\frac{n}{2B}$$
2. normalize so that $\widehat{G}_0 = 1 \pm \delta$

Formally:
$$\widehat{G}_f := \frac{1}{Z}\left(\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U}\right)$$
where Z is a normalization factor.

Upon inspection, $Z = \sum_{f \in [n]} \widehat{H}_f$ works.

Formally:

$$\widehat{G}_f := \frac{1}{Z}\left(\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U}\right)$$

where Z is a normalization factor.

Upon inspection, $Z = \sum_{f\in[n]} \widehat{H}_f$ works.

(Flat region) For any $f \in [-(1-\gamma)\frac{n}{2B}, (1-\gamma)\frac{n}{2B}]$ (flat region) one has

$$\begin{aligned}\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U} &\geq \sum_{f\in[-\gamma\frac{n}{4B}, \gamma\frac{n}{4B}]} \widehat{H}_f \\ &\geq Z - \text{tail of } \widehat{H} \\ &\geq Z - (\delta/n)n \geq Z - \delta\end{aligned}$$

Formally:

$$\widehat{G}_f := \frac{1}{Z}\left(\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U}\right)$$

where Z is a normalization factor.

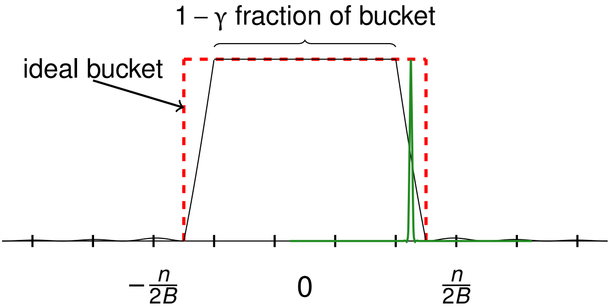
Upon inspection, $Z = \sum_{f \in [n]} \widehat{H}_f$ works.

Indeed, for any $f \notin [-\frac{n}{2B}, \frac{n}{2B}]$ (zero region) one has

$$\widehat{H}_{f-U} + \widehat{H}_{f+1-U} + \ldots + \widehat{H}_{f+U} \leq \sum_{f > \gamma \frac{n}{4B}} \widehat{H}_f$$

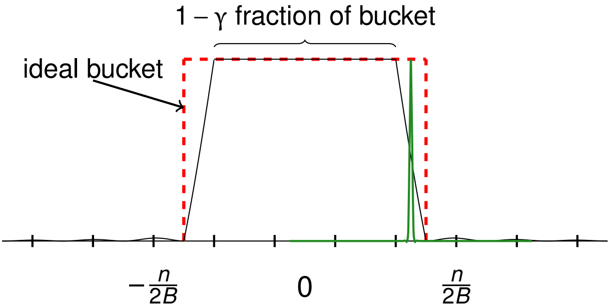
$$\leq \text{tail of } \widehat{H} \leq (\delta/n)n \leq \delta$$

Flat window function



How large is support of $\widehat{G} := \frac{1}{Z}\left(\widehat{H}_{\cdot -U} + \ldots + \widehat{H}_{\cdot +U}\right)$?

Flat window function

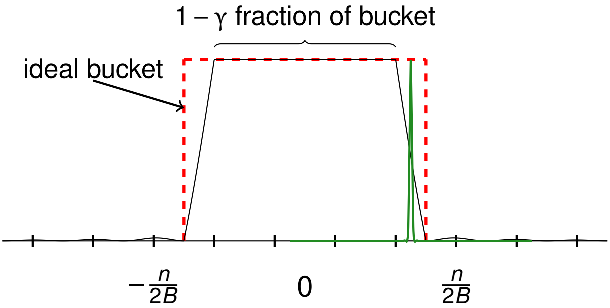


How large is support of $\widehat{G} := \frac{1}{Z}\left(\widehat{H}_{\cdot - U} + \ldots + \widehat{H}_{\cdot + U}\right)$?

By time shift theorem for every $q \in [n]$

$$G_q := H_q \cdot \frac{1}{Z} \sum_{j=-U}^{U} \omega^{qj}$$

Flat window function



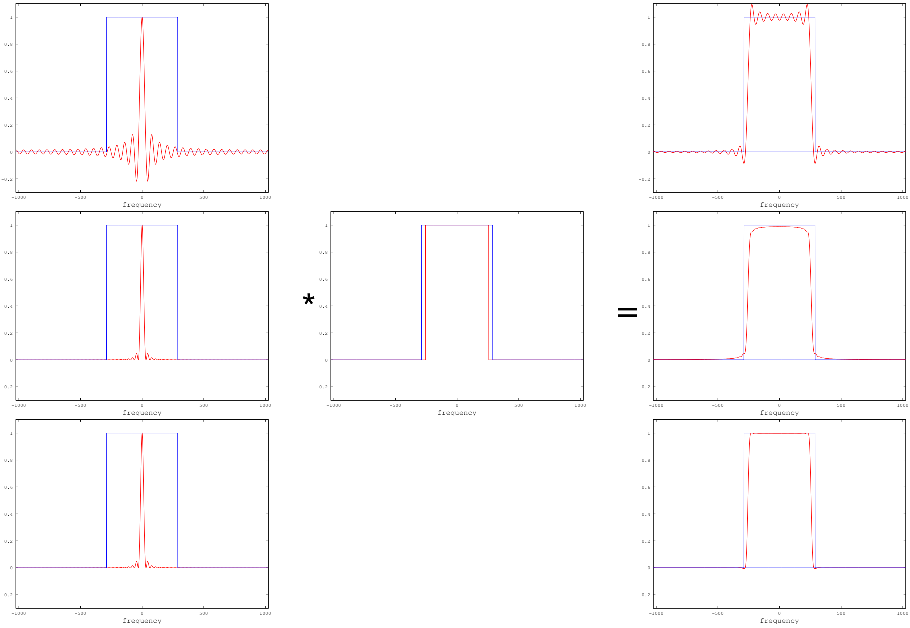
How large is support of $\widehat{G} := \frac{1}{Z}\left(\widehat{H}_{\cdot - U} + \ldots + \widehat{H}_{\cdot + U}\right)$?

By time shift theorem for every $q \in [n]$

$$G_q := H_q \cdot \frac{1}{Z} \sum_{j=-U}^{U} \omega^{qj}$$

Support of G a subset of support of H!

Flat window functions – construction



1. Pseudorandom spectrum permutations
2. Filter construction
3. **Basic block: partial recovery**
4. Full algorithm

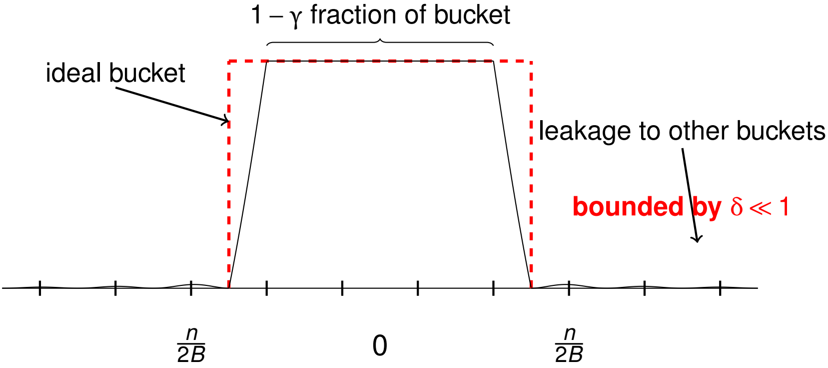
Basic block

Assume

- n is a power of 2
- $\widehat{x}$ contains at most k coefficients with polynomial precision (e.g. $\widehat{x}_f$ in $\{-n^{O(1)}, \ldots, n^{O(1)}\}$)

Then there exists an $O(k \log n)$ time algorithm that

- outputs at most k potential coefficients
- outputs each nonzero $\widehat{x}_f$ correctly with probability at least $1 - \beta$ for a constant $\beta > 0$



Let G be a $(B, \delta/n, \gamma)$-flat window function:

- B buckets
- flat region of width $1-\gamma$
- leakage $\le \delta/n = 1/n^{O(1)}$

Such G can be constructed with

$$\mathrm{supp}(G) = O((k/\gamma)\log n)$$

PartialRecovery – algorithm

Main idea: filter, then run 1-sparse algorithm on each subproblem

PartialRecovery(x, B, γ, δ)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

Run 1-sparse decoding one $\widehat{c}', \widehat{c}''$

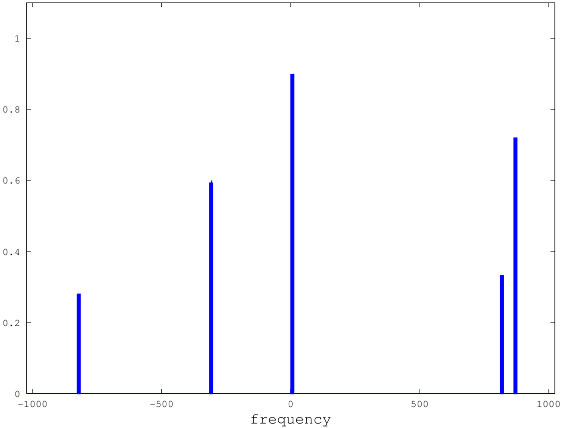
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

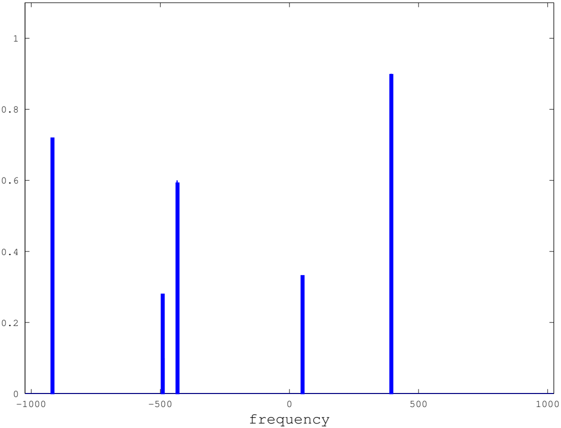
PARTIALRECOVERY – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

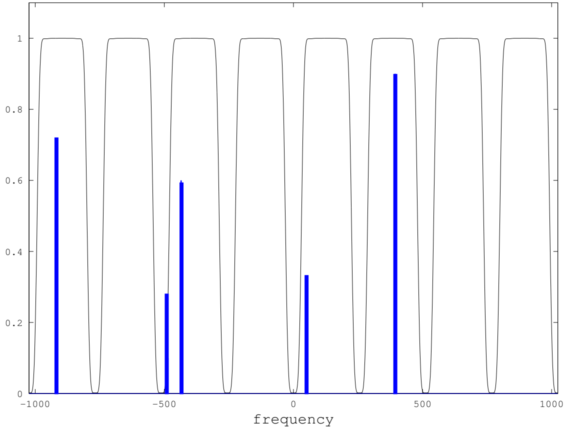
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

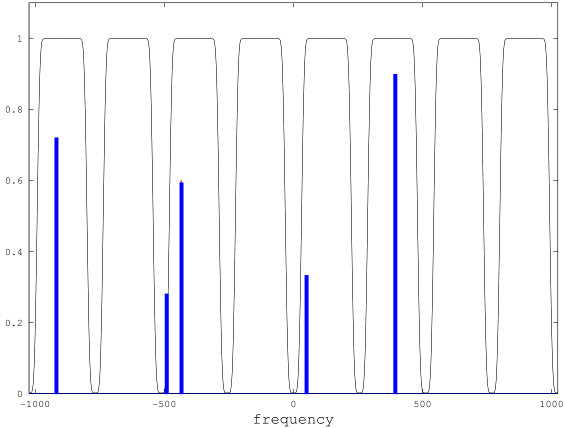
PARTIALRECOVERY – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

PartialRecovery – algorithm

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
If $|\widehat{c}'_{j \cdot n/B}| > 1/2$
Decode from $\widehat{c}'_{j \cdot n/B}, \widehat{c}''_{j \cdot n/B}$
(Two-point sampling)
End
End

Basic block – analysis

Claim

For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Basic block – analysis

Claim

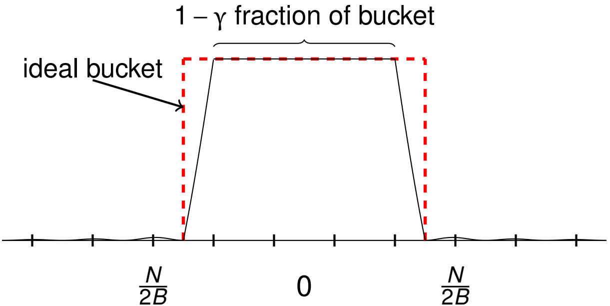
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

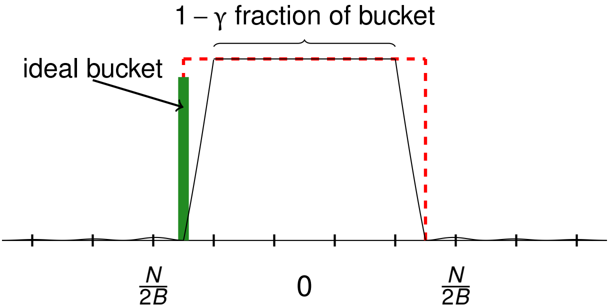
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

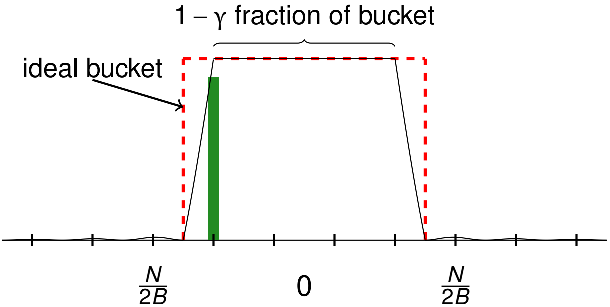
For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

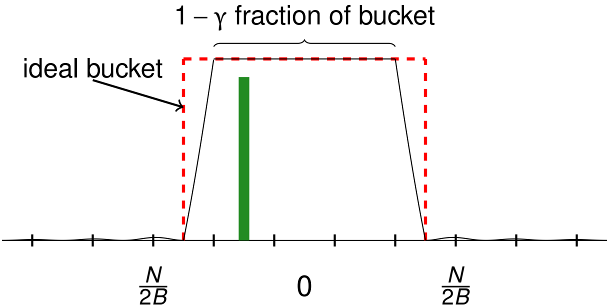
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

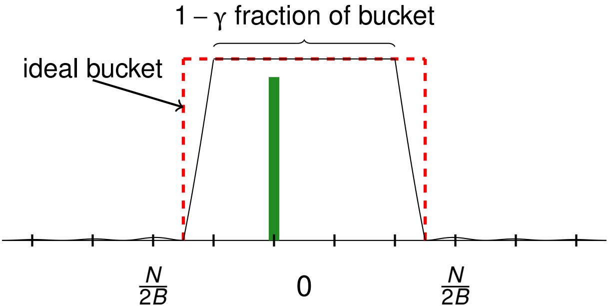
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

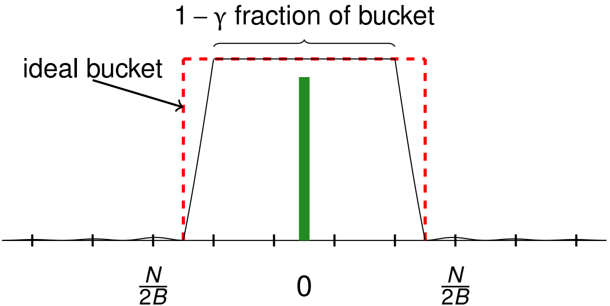
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B + \gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

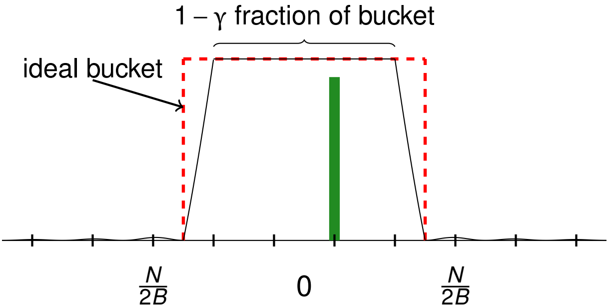
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

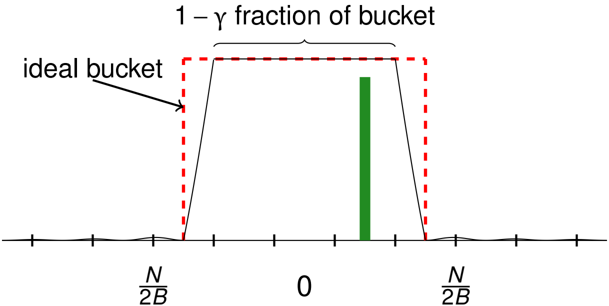
For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

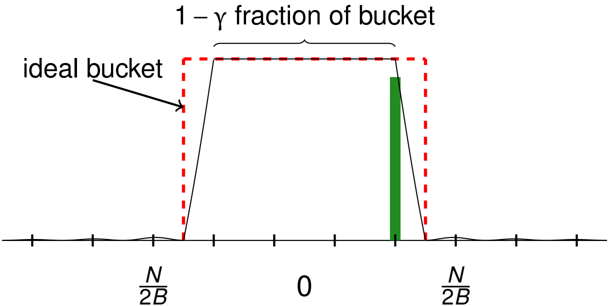
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

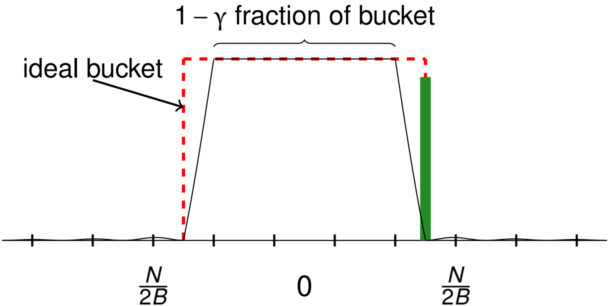
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Computing $\widehat{c}_{j \cdot n/B}$

Option 1 – directly compute FFT of $(x \cdot G)_{-T}, \ldots, (x \cdot G)_T$,
$T = O((k/\gamma) \log n)$

- Can be done in time $O((k/\gamma) \log^2 n)$
- Computes too many samples of $\widehat{x} * \widehat{G}$

Computing $\widehat{c}_{j\cdot n/B}$

Option 1 – directly compute FFT of $(x \cdot G)_{-T}, \ldots, (x \cdot G)_T$, $T = O((k/\gamma)\log n)$

- Can be done in time $O((k/\gamma)\log^2 n)$
- Computes too many samples of $\widehat{x} * \widehat{G}$

Option 2 – alias $x \cdot G$ to B bins first

- Compute
$$b_i = \sum_{j \in [n/B]} x_{i+j\cdot B} G_{i+j\cdot B}$$
- Compute FFT of b in time
$$O(B \log B) = O((k/\gamma)\log n)$$

1. Pseudorandom spectrum permutations
2. Filter construction
3. Basic block: partial recovery
4. **Full algorithm**

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k \quad, \frac{1}{16} \qquad, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k \quad, \frac{1}{16} \qquad, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\mathrm{poly}(n))$

$\ldots$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k \quad, \frac{1}{16} \qquad, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k \quad, \frac{1}{16} \qquad, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k \quad, \frac{1}{16} \qquad, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\mathrm{poly}(n))$

…

Full algorithm

Permute spectrum

Hash to 8 buckets

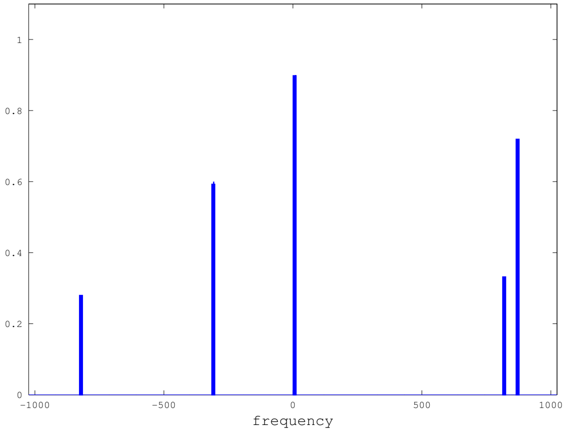
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

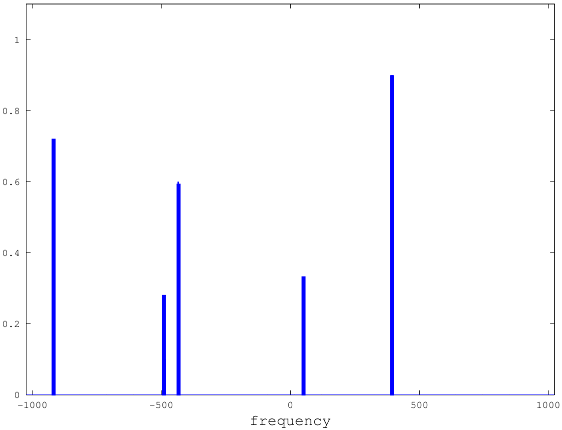
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

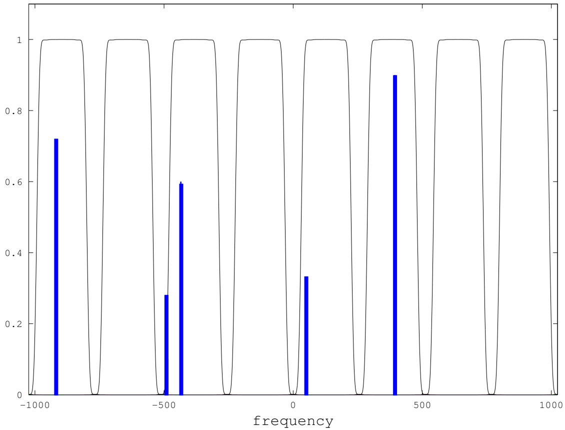
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

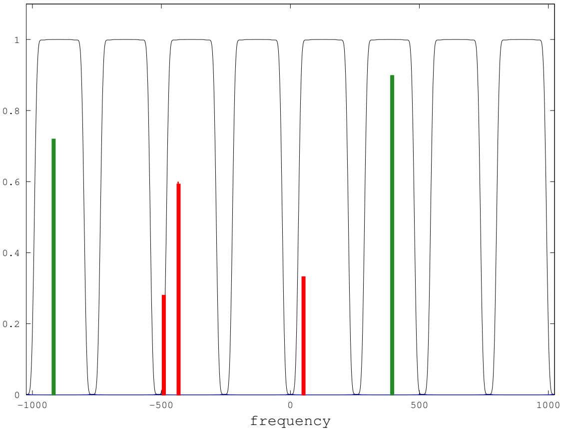
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

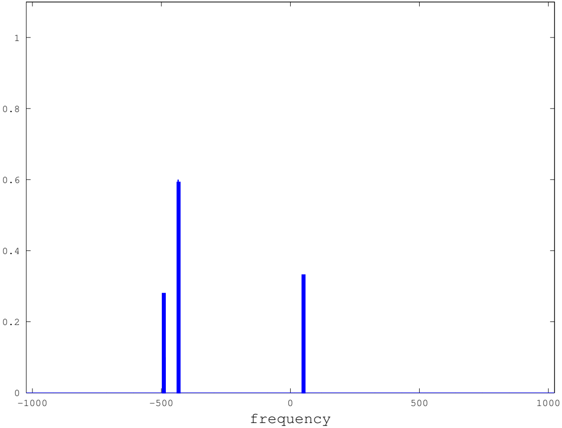
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

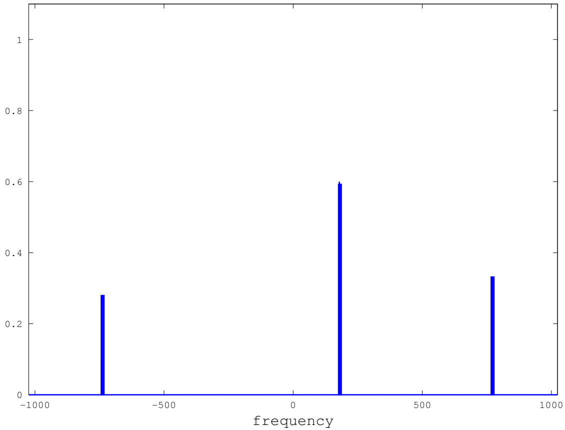
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

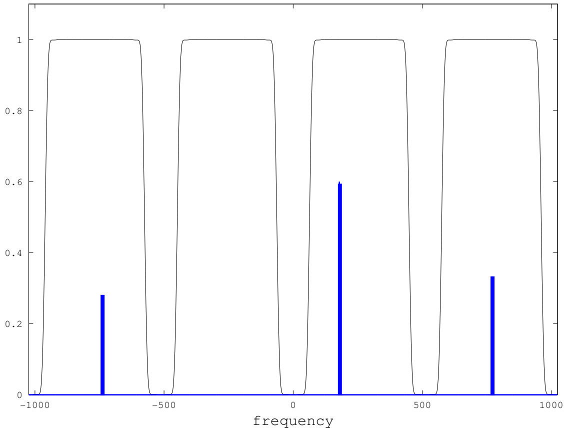
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

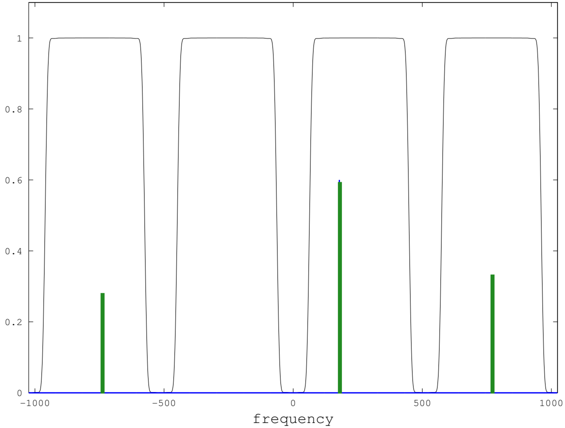
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Modified PartialRecovery

PartialRecovery($B, \alpha, List$)

Choose random b, odd σ

Define $x'_j = x_{\sigma j}\omega^{jb}$
$x''_j = x'_{j+1}$

Compute $\widehat{c}'_{j\cdot\frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j\cdot\frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
  If $|\widehat{c}'_{j\cdot n/B}| > 1/2$
    Decode from $\widehat{c}'_{j\cdot n/B}, \widehat{c}''_{j\cdot n/B}$
    (Two-point sampling)
  End
End

PartialRecovery – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

For each $(pos, val) \in List$

$u \leftarrow \sigma \cdot pos - b$

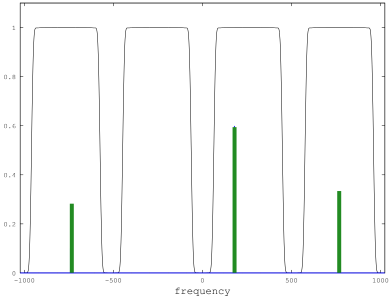
$j \leftarrow$ closest bin to u

$off \leftarrow u - jn/B$

$\widehat{c}'_{j \cdot n/B} \leftarrow \widehat{c}'_{j \cdot n/B} - val \cdot \widehat{G}_{off}$

$\widehat{c}''_{j \cdot n/B} \leftarrow \widehat{c}''_{j \cdot n/B} - val \cdot \omega^{u} \cdot \widehat{G}_{off}$

End



PARTIALRECOVERY – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

For each $(pos, val) \in List$

$u \leftarrow \sigma \cdot pos - b$

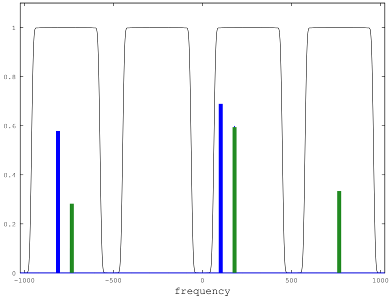
$j \leftarrow$ closest bin to u

$off \leftarrow u - jn/B$

$\widehat{c}'_{j \cdot n/B} \leftarrow \widehat{c}'_{j \cdot n/B} - val \cdot \widehat{G}_{off}$

$\widehat{c}''_{j \cdot n/B} \leftarrow \widehat{c}''_{j \cdot n/B} - val \cdot \omega^{u} \cdot \widehat{G}_{off}$

End



Full algorithm

List $\leftarrow \emptyset$
For $t = 0$ **to** $\log k$
$\quad B_t \leftarrow Ck/4^t$ $\quad \triangleright$ # of buckets to hash to
$\quad \gamma_t \leftarrow 1/(C2^t)$ $\quad \triangleright$ sharpness of filter
$\quad List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$
End

Full algorithm – analysis

Let

$$\widehat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathcal{E}_t$ as

$$\mathcal{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Conditional on $\mathcal{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

Full algorithm – analysis

Let

$$\widehat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Conditional on $\mathscr{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

- probability of collision with another element, which is no more than

$$\frac{k/8^t}{B_t} = \frac{k/8^t}{C \cdot k/4^t} \leq \frac{1}{C \cdot 2^t}$$

Full algorithm – analysis

Let

$$\widehat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Conditional on $\mathscr{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

- probability of collision with another element, which is no more than
$$\frac{k/8^t}{B_t} = \frac{k/8^t}{C \cdot k/4^t} \leq \frac{1}{C \cdot 2^t}$$
- probability of being hashed to the non-flat region, which is no more than
$$O(\gamma_t) = O\left(\frac{1}{C2^t}\right)$$

Full algorithm – analysis

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Then

$$\mathbf{Pr}[\mathscr{E}_t | \mathscr{E}_{t-1}] \leq \mathbf{Pr}[\text{fraction of failures is} \geq 1/8 | \mathscr{E}_{t-1}] \leq O\left(\frac{1}{C \cdot 2^t}\right)$$

Full algorithm – analysis

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \le k/8^t \right\}$$

Then

$$\mathbf{Pr}[\mathscr{E}_t | \mathscr{E}_{t-1}] \le \mathbf{Pr}[\text{fraction of failures is} \ge 1/8 | \mathscr{E}_{t-1}] \le O\left(\frac{1}{C \cdot 2^t}\right)$$

So for a sufficiently large $C > 0$

$$\mathbf{Pr}[\overline{\mathscr{E}}_1 \vee \ldots \vee \overline{\mathscr{E}}_{\log k}] \le O(1/C) \cdot (1/2 + 1/4 + \ldots) = O(1/C) < 1/10$$

Full algorithm – analysis

List $\leftarrow \emptyset$
For $t = 1$ **to** $\log k$
$\quad B_t \leftarrow Ck/4^t$

$\quad \gamma_t \leftarrow 1/(C2^t)$

$\quad List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$
End

Time complexity

- DFT:
  $O(k \log n) + O((k/4)\log n) + \ldots = O(k \log n)$
- List update: $k \cdot \log n$

Sample complexity

List$\leftarrow \emptyset$
For $t=1$ **to** $\log k$
$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \text{PARTIALRECOVERY}(B_t, \gamma_t, List)$
End

Sample complexity $O(k \log n) + O((k/4) \log n) + \ldots = O(k \log n)$

Suboptimal: sufficient to measure $x_0, x_1, \ldots, x_{2k}$ to reconstruct $\widehat{x}$ if $\mathrm{supp}(\widehat{x}) \leq k$ (exercise).

PartialRecovery (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$

$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$

  If $|\widehat{c}'_{j \cdot n/B}| > 1/2$

    Decode from $\widehat{c}'_{j \cdot n/B}, \widehat{c}''_{j \cdot n/B}$

    (Two-point sampling)

  End

End

PartialRecovery (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

For $s = 0, \ldots, \log_2 n$

$r = 1, \ldots, O(\log \log n)$

Define $x_j^{\mathbf{s,0,r}} \leftarrow x_{\sigma(j+\mathbf{r})} \omega^{(j+\mathbf{r})b}$

$x_j^{\mathbf{s,1,r}} \leftarrow x^{s,0,r}_{j+\mathbf{n/2^{s+1}}}$

Compute $(\widehat{x^{s,0,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

$(\widehat{x^{s,1,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

For $j \in [B]$

If $|\widehat{c}'_{j \cdot n/B}| > 1/2$

Decode from $\widehat{x}^{s,0,r}_{j \cdot n/B}$

(As in lecture 1)

(or decode top k elements)

End

End

Runtime and sample complexity

Noiseless: runtime $O(k \log n)$, sample complexity $O(k \log n \log \log n)$

Noisy: runtime $O(k \log^2 n)$, sample complexity $O(k \log^2 n \log \log n)$

$O(\log \log n)$ can be removed, see Hassanieh-Indyk-Katabi-Price'STOC12

Sample complexity lower bound: $\Omega(k \log(n/k))$ (Do Ba, Indyk, Price, Woodruff'SODA10)

Next lecture:

$O(k \log n)$ samples and $O(n \log^3 n)$ runtime
(Indyk-Kapralov'FOCS14)