Sparse Fourier Transform (lecture 2)

Michael Kapralov

\(^1\)IBM Watson

MADALGO’15
Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of $x$:

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

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 $x$:

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

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

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

In last lecture:

- 1-sparse noiseless case: two-point sampling
Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of $x$:

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

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

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

In last lecture:

- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: $O(\log n \log \log \log n)$ time and samples
Given \( x \in \mathbb{C}^n \), compute the Discrete Fourier Transform of \( x \):

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

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

**Goal:** find the top \( k \) coefficients of \( \hat{x} \) approximately

In last lecture:

- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: \( O(\log n \log \log \log n) \) time and samples
- reduction from \( k \)-sparse to 1-sparse case, via filtering
Partition frequency domain into $B \approx k$ buckets

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

$$\hat{u}_j[f] = \begin{cases} \hat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Restricted to a bucket, signal is likely \textit{approximately} 1-sparse!
For each $j = 0, \ldots, B - 1$ let

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

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
\[
u_a^0 = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \hat{\chi}_f \cdot \omega^{f \cdot a}.
\]

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

Then
\[
u_a^0 = (\hat{x} + a * \hat{G})(0)
\]
We want time domain access to $u^0$: for any $a = 0, \ldots, n-1$, compute
\[ u^0_a = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \hat{\chi}_f \cdot \omega^{f \cdot a}. \]

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

Then
\[ u^0_a = (\hat{x}_{-a} \ast \hat{G})(0) \]

For any $j = 0, \ldots, B - 1$
\[ u^j_a = (\hat{x}_{-a} \ast \hat{G})(j \cdot \frac{n}{B}) \]
Reducing $k$-sparse recovery to 1-sparse recovery

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

$$u^j_a = (\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})$$
Reducing $k$-sparse recovery to 1-sparse recovery

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

$$u_j^a = (\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})$$
Reducing $k$-sparse recovery to $1$-sparse recovery

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

$$u^j_a = (\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})$$
Need to evaluate

$$(\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})$$

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

We have access to $x$, not $\hat{x}$. ...
Need to evaluate

\[(\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})\]

for \(j = 0, \ldots, B - 1\).

We have access to \(x\), not \(\hat{x}\)...

By the convolution identity

\[\hat{x} + a \ast \hat{G} = (\hat{x} + a \cdot G)\]
Need to evaluate

\[(\hat{x} + a \ast \hat{G})(j \cdot \frac{n}{B})\]

for \(j = 0, \ldots, B - 1\).

We have access to \(x\), not \(\hat{x}\)...

By the convolution identity

\[\hat{x} + a \ast \hat{G} = (\underbrace{x + a \cdot G}_{\text{suffices to compute}})\]

Suffices to compute

\[x + a \cdot G_{j \cdot \frac{n}{B}}, j = 0, \ldots, B - 1\]
Suffices to compute

\[ x + a \cdot G_j \cdot \frac{n}{B}, j = 0, \ldots, B - 1 \]
Suffices to compute

\[ x \cdot \hat{G}_{j, \frac{n}{B}}, j = 0, \ldots, B - 1 \]
Suffices to compute

\[ \mathcal{X} \cdot G_{j \cdot \frac{n}{B}}, j = 0, \ldots, B - 1 \]

Sample complexity? Runtime?
Suffices to compute

\[ \hat{x} \cdot \hat{G}_{j,n} B_j, j = 0, \ldots, B - 1 \]

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

$$\tilde{x} \cdot \tilde{G}_{j, n_B}, j = 0, \ldots, B - 1$$

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

Design $G$ with $\text{supp}(G) \approx k$ 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 $\rightarrow$ permutation in frequency domain

Claim

Let $\sigma, b \in \mathbb{Z}_n$, $\sigma$ invertible modulo $n$. Let $y_j = x_{\sigma j} \omega^{-jb}$. Then

$$\hat{y}_f = \hat{x}_{\sigma^{-1} (f + b)}.$$ (proof on next slide; a close relative of time shift theorem)

Pseudorandom permutation:

- $\pi$ select $b$ uniformly at random from $\mathbb{Z}_n$.

- $\pi$ select $\sigma$ uniformly at random from $\{1, 3, 5, \ldots, n-1\}$ (invertible numbers modulo $n$).
Pseudorandom spectrum permutations

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

Claim

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

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

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

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

Claim

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

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

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

Pseudorandom permutation:

- select $b$ uniformly at random from $[n]$
- select $\sigma$ 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 \( \hat{y}_f = \hat{x}_{\sigma^{-1}}(f+b) \).

**Proof.**

\[
\hat{y}_f = \frac{1}{n} \sum_{j \in [n]} y_j \omega^{-f \cdot j} \\
= \frac{1}{n} \sum_{j \in [n]} x_{\sigma j} \omega^{-(f+b) \cdot j} \\
= \frac{1}{n} \sum_{i \in [n]} x_i \omega^{-(f+b) \cdot \sigma^{-1} i} \quad \text{(change of variables } i = \sigma j) \\
= \frac{1}{n} \sum_{i \in [n]} x_i \omega^{-\sigma^{-1} (f+b) \cdot i} \\
= \hat{x}_{\sigma^{-1}}(f+b)
\]
Design $G$ with $\text{supp}(G) \approx k$ that approximates rectangular filter?

Our filter $\hat{G}$ will approximate the boxcar. Bound collision probability now.
Partition frequency domain into buckets, permute spectrum

Frequency $i$ collides with frequency $j$ only if $|\sigma_i - \sigma_j| \leq n_B$. 

-1000 -500 0 500 1000

frequency
Partition frequency domain into buckets, permute spectrum
Partition frequency domain into buckets, permute spectrum

Frequency \( i \) collides with frequency \( j \) only if \( |σi − σj| \leq \frac{n}{B} \).
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 $\sigma$ be a uniformly random odd number in $1, 2, \ldots, n$. Then for any $i, j \in [n], i \neq j$ one has

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

Lemma

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

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

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 $O\left( \frac{n}{B2^s} \right)$ 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 $\sigma$ be a uniformly random odd number in $1, 2, \ldots, n$. Then for any $i, j \in [n], i \neq j$ one has

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

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 $O\left(\frac{n}{B2^s}\right)$ 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 $\sigma$ be a uniformly random odd number in $1,2,\ldots,n$. Then for any $i,j \in [n], i \neq j$ one has

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

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 $O\left( \frac{n}{B2^s} \right)$ 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 $\sigma$ be a uniformly random odd number in $1, 2, \ldots, n$. Then for any $i, j \in [n], i \neq j$ one has

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

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 $O\left(\frac{n}{B2^s}\right)$ values of $d'$ that make $\sigma \cdot \Delta$ fall into $[-\frac{n}{B}, \frac{n}{B}]$, out of $n/2^{s+1}$. 
1. Pseudorandom spectrum permutations
2. Filter construction
3. Basic block: partial recovery
4. Full algorithm
Rectangular buckets $\hat{G}$ have full support in time domain...

Approximate rectangular filter with a filter $G$ with small support?

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

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

Have \( \text{supp}(G) = B \approx k \), but buckets leak
In what follows: reduce leakage at the expense of increasing supp($G$)
Window functions

Definition
A symmetric filter $G$ is a \((B, \delta)\)-standard window function if

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

ideal bucket

leakage to other buckets

bounded by $\delta \ll 1$
Window functions

Start with the sinc function:

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

Start with the sinc function:

\[
\hat{G}_f := \frac{\sin(\pi(B+1)f/n)}{(B+1) \cdot \pi f/n}
\]

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

\[
|\hat{G}_f| \leq \frac{1}{(B+1)\pi f/n} \leq \frac{1}{\pi/2} \leq 2/\pi \leq 0.9
\]
Consider **powers of the sinc function**:

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

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

\[ |\hat{G}_f|^r \leq (0.9)^r \]
Consider \textbf{powers of the sinc function}:

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

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

\[
|\hat{G}_f|^r \leq (0.9)^r
\]
Window functions

Consider **powers of the sinc function**: $\hat{G}_f^r$

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

$$|\hat{G}_f|^r \leq (0.9)^r$$
Consider powers of the sinc function: $\hat{G}_f^r$.

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

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

So setting $r = O(\log(1/\delta))$ is sufficient!
Window functions

Definition
A symmetric filter $G$ is a $(B, \delta)$-standard window function if

1. $\hat{G}_0 = 1$
2. $\hat{G}_f \geq 0$
3. $|\hat{G}_f| \leq \delta$ for $f \not\in \left[-\frac{n}{2B}, \frac{n}{2B}\right]$

ideal bucket

leakage to other buckets

bounded by $\delta \ll 1$
Window functions

**Definition**

A symmetric filter $G$ is a $(B, \delta)$-standard window function if

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

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

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

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

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

By the convolution identity \( G^r = G^0 \ast G^0 \ast \ldots \ast G^0 \)
Let

\[ G_j := \begin{cases} 
\frac{1}{B+1} & \text{if } j \in \left[-\frac{B}{2}, \frac{B}{2}\right] \\
0 & \text{o.w.}
\end{cases} \]

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

By the convolution identity \( G^r = G^0 \ast G^0 \ast \ldots \ast G^0 \)

Support of \( G^0 \) is in \([-B/2, B/2]\), so

\[ \text{supp}(G \ast \ldots \ast G) \subseteq [-r \cdot B/2, r \cdot B/2] \]
Let  
\[ G_j := \begin{cases} 
\frac{1}{B+1} & \text{if } j \in [-B/2, B/2] \\
0 & \text{o.w.} 
\end{cases} \]

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

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

By the convolution identity \( G' = G^0 \ast G^0 \ast \ldots \ast G^0 \)

Support of \( G^0 \) is in \([-B/2, B/2]\), so

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

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

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

By the convolution identity \( G' = G^0 \ast G^0 \ast \ldots \ast G^0 \)

Support of \( G^0 \) is in \([-B/2, B/2]\), so

\[ \text{supp}(G \ast \ldots \ast G) \subseteq [-r \cdot B/2, r \cdot B/2] \]
Let
\[ G_j := \begin{cases} 
\frac{1}{(B + 1)} & \text{if } j \in [-B/2, B/2] \\
0 & \text{o.w.}
\end{cases} \]

Let \( \hat{G}' := (\hat{G^0})' \). How large is the support of \( G' \)?

By the convolution identity \( G' = G^0 * G^0 * \ldots * G^0 \)

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

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

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

By the convolution identity \( G' = G^0 \ast G^0 \ast \ldots \ast G^0 \)

Support of \( G^0 \) is in \([-B/2, B/2]\), so

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

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

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

By the convolution identity \( G^r = G^0 \ast G^0 \ast \ldots \ast G^0 \)

Support of \( G^0 \) is in \([-B/2, B/2]\), so

\[ \text{supp}(G \ast \ldots \ast G) \subseteq [-r \cdot B/2, r \cdot B/2] \]
Flat window function

Definition

A symmetric filter $G$ is a $(B, \delta, \gamma)$-flat window function if

1. $\hat{G}_j \geq 1 - \delta$ for all $j \in \left[-(1 - \gamma)\frac{n}{2B}, (1 - \gamma)\frac{n}{2B}\right]$

2. $\hat{G}_j \in [0, 1]$ for all $j$

3. $|\hat{G}_f| \leq \delta$ for $f \not\in \left[-\frac{n}{2B}, \frac{n}{2B}\right]$
Flat window function

Definition

A symmetric filter $G$ is a $(B, \delta, \gamma)$-flat window function if

1. $\hat{G}_j \geq 1 - \delta$ for all $j \in \left[-(1 - \gamma) \frac{n}{2B}, (1 - \gamma) \frac{n}{2B}\right]$
2. $\hat{G}_j \in [0, 1]$ for all $j$
3. $|\hat{G}_f| \leq \delta$ for $f \not\in \left[-\frac{n}{2B}, \frac{n}{2B}\right]$

$1 - \gamma$ fraction of bucket

ideal bucket
Flat window function

Definition
A symmetric filter $G$ is a $(B, \delta, \gamma)$-flat window function if

1. $\hat{G}_j \geq 1 - \delta$ for all $j \in \left[-(1 - \gamma)\frac{n}{2B}, (1 - \gamma)\frac{n}{2B}\right]$
2. $\hat{G}_j \in [0, 1]$ for all $j$
3. $|\hat{G}_f| \leq \delta$ for $f \notin \left[-\frac{n}{2B}, \frac{n}{2B}\right]$
Let $H$ be a $(2B/\gamma, \delta/n)$-standard window function. Note that

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

for all $f$ outside of

$$\left[-\gamma \frac{n}{4B}, \gamma \frac{n}{4B}\right].$$
Let $H$ be a $(2B/\gamma, \delta/n)$-standard window function. Note that $|\hat{H}_f| \leq \delta/n$ for all $f$ outside of $\left[ -\gamma \frac{n}{4B}, \gamma \frac{n}{4B} \right]$. 
Let $H$ be a $(2B/\gamma, \delta/n)$-standard window function. Note that

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

for all $f$ outside of

$$\left[-\gamma \frac{n}{4B}, \gamma \frac{n}{4B}\right].$$
Let $H$ be a $(2B/\gamma, \delta/n)$-standard window function. Note that

$$|\hat{H}_f| \leq \frac{\delta}{n}$$

for all $f$ outside of

$$\left[-\gamma \frac{n}{4B}, \gamma \frac{n}{4B}\right].$$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

1. sum up shifts $\hat{H}_{-\Delta}$ over all $\Delta \in [-U, U]$, where
   \[
   U = (1 - \gamma / 2) \frac{n}{2B}
   \]

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

1. sum up shifts $\hat{H}_{-\Delta}$ over all $\Delta \in [-U, U]$, where
   \[ U = (1 - \gamma/2) \frac{n}{2B} \]

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct \( \hat{G} \):

1. sum up shifts \( \hat{H}_{-\Delta} \) over all \( \Delta \in [-U, U] \), where
   \[
   U = (1 - \gamma / 2) \frac{n}{2B}
   \]

2. normalize so that \( \hat{G}_0 = 1 \pm \delta \)
Flat window function – construction

To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
Flat window function – construction

To construct $\hat{G}$:

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

To construct $\hat{G}$:

1. sum up shifts $\hat{H}_{-\Delta}$ over all $\Delta \in [-U, U]$, where
   \[ U = (1 - \gamma/2) \frac{n}{2B} \]

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
To construct $\hat{G}$:

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

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

2. normalize so that $\hat{G}_0 = 1 \pm \delta$
To construct $\hat{G}$:

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

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

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

Formally:

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

where $Z$ is a normalization factor.
To construct $\hat{G}$:

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

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

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

Formally:

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

where $Z$ is a normalization factor.

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

\[ \hat{G}_f := \frac{1}{Z} \left( \hat{H}_{f-U} + \hat{H}_{f+1-U} + \cdots + \hat{H}_{f+U} \right) \]

where \( Z \) is a normalization factor.

Upon inspection, \( Z = \sum_{f \in [n]} \hat{H}_f \) works.

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

\[ \hat{H}_{f-U} + \hat{H}_{f+1-U} + \cdots + \hat{H}_{f+U} \geq \sum_{f \in \left[ -\gamma \frac{n}{4B}, \gamma \frac{n}{4B} \right]} \hat{H}_f \]

\[ \geq Z - \text{tail of } \hat{H} \]

\[ \geq Z - \left( \frac{\delta}{n} \right) n \geq Z - \delta \]
Formally:
\[
\hat{G}_f := \frac{1}{Z} \left( \hat{H}_{f-U} + \hat{H}_{f+1-U} + \ldots + \hat{H}_{f+U} \right)
\]

where \( Z \) is a normalization factor.

Upon inspection, \( Z = \sum_{f \in [n]} \hat{H}_f \) works.

Indeed, for any \( f \not\in [-\frac{n}{2B}, \frac{n}{2B}] \) (zero region) one has
\[
\hat{H}_{f-U} + \hat{H}_{f+1-U} + \ldots + \hat{H}_{f+U} \leq \sum_{f > \gamma \frac{n}{4B}} \hat{H}_f \\
\leq \text{tail of } \hat{H} \leq (\delta/n)n \leq \delta
\]
Flat window function

1 – γ fraction of bucket

ideal bucket

How large is support of \( \hat{G} := \frac{1}{Z} \left( \hat{H}_- U + \ldots + \hat{H}_+ U \right) \)?
Flat window function

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

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

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

- $\hat{x}$ contains at most $k$ coefficients with polynomial precision (e.g. $\hat{x}_f$ 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 $\hat{x}_f$ 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 $\leq \delta/n = 1/n^{O(1)}$

Such $G$ can be constructed with

$$\text{supp}(G) = O((k/\gamma) \log n)$$
**PARTIAL RECOVERY – algorithm**

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

**PARTIAL RECOVERY**($x, B, \gamma, \delta$)

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

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

$x''_j \leftarrow x'_j + 1$

Compute $\hat{c}'_j \cdot n_B, j \in [B], \text{ where } c' = x' \cdot G$

$\hat{c}''_j \cdot n_B, j \in [B], \text{ where } c'' = x'' \cdot G$

Run 1-sparse decoding one $\hat{c}'$, $\hat{c}''$
PARTIAL RECOVERY – algorithm

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

Permute spectrum
Filter signal
1-sparse decoding

Isolated frequencies are decoded successfully
PARTIAL RECOVERY – algorithm

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

Permute spectrum
Filter signal
1-sparse decoding

Isolated frequencies are decoded successfully
Recovering 5-sparse signal $\hat{x}$ from measurements of $x$

- Permute spectrum
- Filter signal
- 1-sparse decoding

Isolated frequencies are decoded successfully
PARTIAL RECOVERY – algorithm

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

Permute spectrum

Filter signal

1-sparse decoding

Isolated frequencies are decoded successfully
PARTIAL \textsc{Recovery} – algorithm

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

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

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

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

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

\textbf{For} $j \in [B]$

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

Decode from $\hat{c}'_{j \cdot n \over B}, \hat{c}''_{j \cdot n \over B}$

\hspace{1cm} (Two-point sampling)

\textbf{End}

\textbf{End}
Basic block – analysis

Claim
For each $u \in \text{supp}(\hat{x})$ 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 $O(\gamma)$

$N^2/B$

Ideal bucket

$1 - \gamma$

Fraction of bucket

$57 / 75$
Basic block – analysis

Claim
For each $u \in \text{supp}(\hat{x})$ 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 \( u \in \text{supp}(\hat{x}) \) 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 $u \in \text{supp}(\hat{x})$ 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 \( u \in \text{supp}(\hat{x}) \) 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 \( u \in \text{supp}(\hat{x}) \) 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 \( O(\gamma) \)

\[ \text{Ideal bucket} \]
\[ 1 - \gamma \text{ fraction of bucket} \]
Basic block – analysis

Claim
For each \( u \in \text{supp}(\hat{x}) \) 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 \( u \in \text{supp}(\hat{x}) \) 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 $u \in \text{supp}(\hat{x})$ 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 \( u \in \text{supp}(\hat{x}) \) 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 \( u \in \text{supp}(\hat{x}) \) 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 \( O(\gamma) \)
Computing $\hat{c}_{j \cdot n/B}$

**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 $\hat{x} \ast \hat{G}$
Computing $\hat{c}_{j \cdot n/B}$

**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 $\hat{x} \ast \hat{G}$

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

- Compute

$$b_i = \sum_{j \in \lfloor n/B \rfloor} x_{i+j \cdot B} G_{i+j \cdot B}$$

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

$\text{PARTIALRECOVERY}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$
Let $C > 0$ be a sufficiently large constant.

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

\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/poly(n))
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))
Let $C > 0$ be a sufficiently large constant.

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

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

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

$\text{PARTIALRECOVERY}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/poly(n))$
Let $C > 0$ be a sufficiently large constant.

**PARTIAL\_RECOVERY**($x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n)$)

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

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

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

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

\[
\text{PARTIAL\,RECOVERY}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n))
\]
Let $C > 0$ be a sufficiently large constant.

\[
\text{PARTIALRECOVERY}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n))
\]

\[
\text{PARTIALRECOVERY}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))
\]
Full algorithm

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

\[
\text{PARTIAL\textsc{Recovery}}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}\,(n))
\]

\[
\text{PARTIAL\textsc{Recovery}}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}\,(n))
\]

\[
\text{PARTIAL\textsc{Recovery}}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}\,(n))
\]
Let $C > 0$ be a sufficiently large constant.

\[ \text{PARTIAL\textsc{RECOVERY}}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n)) \]

\[ \text{PARTIAL\textsc{RECOVERY}}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n)) \]

\[ \text{PARTIAL\textsc{RECOVERY}}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n)) \]

\[ \text{PARTIAL\textsc{RECOVERY}}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n)) \]
Let $C > 0$ be a sufficiently large constant.

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

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

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

\textsc{PartialRecovery}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{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

\[ \frac{63}{75} \]
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 **PARTIAL RECOVERY**

**PARTIAL RECOVERY**($B, \alpha, \text{List}$)

Choose random $b$, odd $\sigma$

Define $x'_j = x_{\sigma j} \omega^j b$

$x''_j = x'_{j+1}$

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

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

For $j \in [B]$

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

Decode from $\hat{c}'_{j \cdot \frac{n}{B}}, \hat{c}''_{j \cdot \frac{n}{B}}$

(Two-point sampling)

End

End
PARTIAL RECOVERY – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

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

\[ u \leftarrow \sigma \cdot pos - b \]

\[ j \leftarrow \text{closest bin to } u \]

\[ \text{off} \leftarrow u - \text{jn}/B \]

\[ \hat{c}_{j \cdot n/B}^{I} \leftarrow \hat{c}_{j \cdot n/B}^{I} - val \cdot \hat{G}_{\text{off}} \]

\[ \hat{c}_{j \cdot n/B}^{II} \leftarrow \hat{c}_{j \cdot n/B}^{II} - val \cdot \omega_{u} \cdot \hat{G}_{\text{off}} \]

End
PARTIAL RECOVERY – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

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

\[ u \leftarrow \sigma \cdot pos - b \]

\[ j \leftarrow \text{closest bin to } u \]

\[ \text{off} \leftarrow u - jn/B \]

\[ \hat{c}' \cdot n/B \rangle \leftarrow \hat{c}' \cdot n/B \rangle - val \cdot \hat{G}_{\text{off}} \]

\[ \hat{c}'' \cdot n/B \rangle \leftarrow \hat{c}'' \cdot n/B \rangle - val \cdot \omega^u \cdot \hat{G}_{\text{off}} \]

End
Full algorithm

List ← φ
For $t = 0$ to $\log k$
  $B_t ← Ck/4^t$  ▷ # of buckets to hash to
  $\gamma_t ← 1/(C2^t)$  ▷ sharpness of filter
  $List ← List + \text{PARTIALRECOVERY}(B_t, \gamma_t, List)$
End
Full algorithm – analysis

Let

\[ \hat{e}_t \leftarrow \text{contents of the list after stage } t. \]

Define ‘good event’ \( \mathcal{E}_t \) as

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

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

\[ \hat{e}_t \leftarrow \text{contents of the list after stage } t. \]

Define ‘good event’ \( E_t \) as

\[ E_t := \{ \| \hat{x} - \hat{e}_t \|_0 \leq k/8^t \} \]

Conditional on \( E_{t-1} \), 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

\[ \hat{e}_t \leftarrow \text{contents of the list after stage } t. \]

Define ‘good event’ \( E_t \) as

\[ E_t := \left\{ ||\hat{x} - \hat{e}_t||_0 \leq k/8^t \right\} \]

Conditional on \( E_{t-1} \), 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

\[ O(\gamma_t) = O\left( \frac{1}{C2^t} \right) \]
Define ‘good event’ $\mathcal{E}_t$ as

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

Then

$$
\Pr[\mathcal{E}_t | \mathcal{E}_{t-1}] \leq \Pr[\text{fraction of failures is } \geq 1/8 | \mathcal{E}_{t-1}] \leq O\left(\frac{1}{C \cdot 2^t}\right)
$$
Define ‘good event’ $\mathcal{E}_t$ as

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

Then

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

So for a sufficiently large $C > 0$

$$\Pr[\overline{\mathcal{E}}_1 \lor \ldots \lor \overline{\mathcal{E}}_{\log k}] \leq O(1/C) \cdot (1/2 + 1/4 + \ldots) = O(1/C) < 1/10$$
Full algorithm – analysis

List ← ∅
For $t = 1$ to $\log k$
  $B_t ← Ck/4^t$
  $\gamma_t ← 1/(C2^t)$
  List ← List + PARTIALRECOVERY($B_t, \gamma_t, \text{List}$)
End

Time complexity

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

\[
\text{List} \leftarrow \emptyset \\
\text{For } t = 1 \text{ to } \log k \\
\quad B_t \leftarrow Ck/4^t \\
\quad \gamma_t \leftarrow 1/(C2^t) \\
\quad \text{List} \leftarrow \text{List} + \text{PARTIALRECOVERY}(B_t, \gamma_t, \text{List}) \\
\text{End}
\]

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

\textbf{Suboptimal:} sufficient to measure \( x_0, x_1, \ldots, x_{2k} \) to reconstruct \( \hat{x} \) if \( \text{supp}(\hat{x}) \leq k \) (exercise).
PARTIAL RECOVERY (noisy setting)

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

Define \( x_j' \leftarrow x_{\sigma j} \omega^{jb} \) \n\[
    x_j'' \leftarrow x_{j+1}'
\]

Compute \( \hat{c}_{j,n/B}' \), \( j \in [B] \), where \( c' = x' \cdot G \)

\[
\hat{c}_{j,n/B}'' \), \( j \in [B] \), where \( c'' = x'' \cdot G \)

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

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

Define $x_j^{s,0,r} \leftarrow x_{\sigma(j+r)} \omega^{(j+r)b}$
$x_j^{s,1,r} \leftarrow x_j^{s,0,r} j+n/2^{s+1}$

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

For $j \in [B]$
If $|\hat{c}_{j \cdot n/B}| > 1/2$
Decode from $\hat{x}^{s,0,r}_{j \cdot n/B}$
(As in lecture 1)
End

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

$\sigma \in \{1, 2, \ldots, n\}$

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

(As in lecture 1)
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: $\Omega(k \log(\frac{n}{k}))$ (Do Ba, Indyk, Price, Woodruff’SODA10)
Next lecture:

$O(k \log n)$ samples and $O(n \log^3 n)$ runtime

(Indyk-Kapralov'FOCS14)