A Faster Fourier Transform on Sparse Data
and Beyond

Eric Price

MIT
The Fourier Transform
Conversion between time and frequency domains

Time Domain

Frequency Domain

Fourier Transform

Displacement of Air

Concert A
The Fourier Transform is Ubiquitous

- Audio
- Video
- Medical Imaging
- Radar
- GPS
- Oil Exploration
Computing the Discrete Fourier Transform

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

Naive multiplication: $O(n^2)$.

Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

The method greatly reduces the tediousness of mechanical calculations. – Carl Friedrich Gauss, 1805

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

Can we do better?

When can we compute the Fourier Transform in sublinear time?

Eric Price (MIT)

A Faster Fourier Transform on Sparse Data
Computing the Discrete Fourier Transform

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

Fast Fourier Transform: $O(n \log n)$ time. [Cooley-Tukey, 1965]

- The method greatly reduces the tediousness of mechanical calculations. – Carl Friedrich Gauss, 1805
- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]

Can we do better?

When can we compute the Fourier Transform in sublinear time?
Computing the Discrete Fourier Transform

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

The method greatly reduces the tediousness of mechanical calculations. – Carl Friedrich Gauss, 1805

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

Can we do better?

When can we compute the Fourier Transform in sublinear time?

Eric Price (MIT)
Computing the Discrete Fourier Transform

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

'[T]he method greatly reduces the tediousness of mechanical calculations.'

– Carl Friedrich Gauss, 1805
Computing the Discrete Fourier Transform

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

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

- Carl Friedrich Gauss, 1805

- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]
Computing the Discrete Fourier Transform

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

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

– Carl Friedrich Gauss, 1805

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

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

[The method greatly reduces the tediousness of mechanical calculations.]

– Carl Friedrich Gauss, 1805

- By hand: $22n \log n$ seconds. [Danielson-Lanczos, 1942]
- Can we do much better?
Computing the Discrete Fourier Transform

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

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

– Carl Friedrich Gauss, 1805

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

When can we compute the Fourier Transform in sublinear time?
Idea: Leverage Sparsity

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

Time Signal

Frequency (Exactly sparse)

Frequency (Approximately sparse)
Idea: Leverage Sparsity

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

- **Time Signal**
- **Frequency** (Exactly sparse)
- **Frequency** (Approximately sparse)

Sparsity is common:

- Audio
- Video
- Medical Imaging
- Radar
- GPS
- Oil Exploration
Idea: Leverage *Sparsity*

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

- **Time Signal**
- **Frequency** (Exactly sparse)
- **Frequency** (Approximately sparse)

Sparsity is common:

**Goal of this work:** a *sparse* Fourier transform

*Faster* Fourier Transform on sparse data.
Sparse Fourier Transform

- Overview
- Technical Details
Talk Outline

1. Sparse Fourier Transform
   - Overview
   - Technical Details

2. Beyond: Sparse Recovery / Compressive Sensing
   - Overview
   - Adaptivity
   - Conclusion
1. Sparse Fourier Transform
   - Overview
   - Technical Details

2. Beyond: Sparse Recovery / Compressive Sensing
   - Overview
   - Adaptivity
   - Conclusion
My Contributions

Goal: Compute the Fourier transform $\hat{x} = Fx$ when $\hat{x}$ is $k$-sparse.

- Theory:
  - The fastest algorithm for Fourier transforms of sparse data.
  - The only algorithms faster than FFT for all $k = o(n)$. 

Eric Price (MIT)
My Contributions

Goal: Compute the Fourier transform $\hat{x} = Fx$ when $\hat{x}$ is $k$-sparse.

Theory:
- The fastest algorithm for Fourier transforms of sparse data.
- The only algorithms faster than FFT for all $k = o(n)$.

Practice:
- Implementation is faster than FFTW for a wide range of inputs.
- Orders of magnitude faster than previous sparse Fourier transforms.
- Useful in multiple applications.
Applications of ideas

http://groups.csail.mit.edu/netmit/sFFT/workshop.html

- GPS [HAKI]: 2× faster
Applications of ideas

http://groups.csail.mit.edu/netmit/sFFT/workshop.html

- GPS [HAKI]: 2× faster
- Spectrum sensing [HSAHK]: 6× lower sampling rate
Applications of ideas

http://groups.csail.mit.edu/netmit/sFFT/workshop.html

- GPS [HAKI]: $2 \times$ faster
- Spectrum sensing [HSAHK]: $6 \times$ lower sampling rate
- Dense FFT over clusters [TPKP]: $2 \times$ faster
Applications of ideas

http://groups.csail.mit.edu/netmit/sFFT/workshop.html

- GPS [HAKI]: 2× faster
- Spectrum sensing [HSAHK]: 6× lower sampling rate
- Dense FFT over clusters [TPKP]: 2× faster
- ...

Eric Price (MIT)
Talk Outline

1. Sparse Fourier Transform
   - Overview
   - Technical Details

2. Beyond: Sparse Recovery / Compressive Sensing
   - Overview
   - Adaptivity
   - Conclusion
Theoretical Results
For a signal of size $n$ with $k$ large frequencies

- Prior work [KM92, GL89, Mansour ’92, GGIMS02, AGS03, GMS05, Iwen ’10, Akavia ’10]
  - All take at least $k \log^4 n$ time.
  - Only better than FFT if $k \ll n/\log^3 n$. 

Eric Price (MIT)
A Faster Fourier Transform on Sparse Data
Theoretical Results
For a signal of size \( n \) with \( k \) large frequencies

- Prior work [KM92, GL89, Mansour ’92, GGIMS02, AGS03, GMS05, Iwen ’10, Akavia ’10]
  - All take at least \( k \log^4 n \) time.
  - Only better than FFT if \( k \ll n / \log^3 n \).

- Our results [HIKP12a, HIKP12b]
Theoretical Results
For a signal of size $n$ with $k$ large frequencies

- Prior work [KM92, GL89, Mansour ’92, GGIMS02, AGS03, GMS05, Iwen ’10, Akavia ’10]
  - All take at least $k \log^4 n$ time.
  - Only better than FFT if $k \ll n/\log^3 n$.

- Our results [HIKP12a, HIKP12b]
  - Exactly $k$-sparse: $O(k \log n)$
    - Optimal if FFT is optimal.
Theoretical Results
For a signal of size $n$ with $k$ large frequencies

- Prior work [KM92, GL89, Mansour ’92, GGIMS02, AGS03, GMS05, Iwen ’10, Akavia ’10]
  - All take at least $k \log^4 n$ time.
  - Only better than FFT if $k \ll n/\log^3 n$.

- Our results [HIKP12a, HIKP12b]
  - Exactly $k$-sparse: $O(k \log n)$
    - Optimal if FFT is optimal.
  - Approximately $k$-sparse: $O(k \log(n/k) \log n)$

\[
\|\text{result} - \hat{x}\|_2 \leq (1 + \epsilon) \min_{k\text{-sparse } \hat{x}(k)} \|\hat{x}(k) - \hat{x}\|_2
\]
Theoretical Results
For a signal of size $n$ with $k$ large frequencies

- Prior work [KM92, GL89, Mansour ’92, GGIMS02, AGS03, GMS05, Iwen ’10, Akavia ’10]
  - All take at least $k \log^4 n$ time.
  - Only better than FFT if $k \ll n / \log^3 n$.

- Our results [HIKP12a, HIKP12b]
  - Exactly $k$-sparse: $O(k \log n)$
    - Optimal if FFT is optimal.
  - Approximately $k$-sparse: $O(k \log(n/k) \log n)$

\[
\|\text{result} - \hat{x}\|_2 \leq (1 + \epsilon) \min_{k\text{-sparse } \hat{x}(k)} \|\hat{x}(k) - \hat{x}\|_2
\]

- Better than FFT for any $k = o(n)$
Discrete Fourier Transform (DFT) Definition

- Given $x \in \mathbb{C}^n$, compute Fourier transform $\hat{x}$:

$$
\hat{x}_j = \frac{1}{n} \sum_j \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i/n}
$$
Discrete Fourier Transform (DFT) Definition

- Given $x \in \mathbb{C}^n$, compute Fourier transform $\hat{x}$:

$$\hat{x}_i = \frac{1}{n} \sum_j \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i/n}$$

$$\hat{x} = F x \quad \text{for} \quad F_{ij} = \omega^{-ij}/n$$
Discrete Fourier Transform (DFT) Definition

- Given $x \in \mathbb{C}^n$, compute Fourier transform $\hat{x}$:

  $$
  \hat{x}_j = \frac{1}{n} \sum_{j} \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i / n}
  $$

  $$
  \hat{x} = F x \quad \text{for} \quad F_{ij} = \omega^{-ij} / n
  $$

- Inverse transform almost identical:
Discrete Fourier Transform (DFT) Definition

- Given $x \in \mathbb{C}^n$, compute Fourier transform $\hat{x}$:
  \[
  \hat{x}_i = \frac{1}{n} \sum_j \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i / n}
  \]
  \[
  \hat{x} = Fx \quad \text{for} \quad F_{ij} = \omega^{-ij} / n
  \]

- Inverse transform almost identical:
  \[
  x_i = \sum_j \omega^{ij} \hat{x}_j
  \]
  \[
  \omega \rightarrow \omega^{-1}, \text{ scale}
  \]
Discrete Fourier Transform (DFT) Definition

- Given $x \in \mathbb{C}^n$, compute Fourier transform $\hat{x}$:

  $$
  \hat{x}_i = \frac{1}{n} \sum_j \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i / n}
  $$

  $$
  \hat{x} = F x \quad \text{for} \quad F_{ij} = \omega^{-ij} / n
  $$

- Inverse transform almost identical:

  $$
  x_i = \sum_j \omega^{ij} \hat{x}_j
  $$

  $\omega \rightarrow \omega^{-1}$, scale

- Lots of nice properties
Discrete Fourier Transform (DFT) Definition

- Given \( x \in \mathbb{C}^n \), compute Fourier transform \( \hat{x} \):

\[
\hat{x}_i = \frac{1}{n} \sum_j \omega^{-ij} x_j \quad \text{for} \quad \omega = e^{2\pi i / n}
\]

\[
\hat{x} = F x \quad \text{for} \quad F_{ij} = \omega^{-ij} / n
\]

- Inverse transform almost identical:

\[
x_i = \sum_j \omega^{ij} \hat{x}_j
\]

  - \( \omega \to \omega^{-1} \), scale

- Lots of nice properties

  - Convolution \( \leftrightarrow \) Multiplication
Algorithm

Simpler case: \(\hat{x}\) is exactly \(k\)-sparse.
Algorithm

Simpler case: $\hat{x}$ is exactly $k$-sparse.

**Theorem**

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

Simpler case: \( \hat{x} \) is exactly \( k \)-sparse.

Theorem

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

Still kind of hard.
Algorithm

Simpler case: $\hat{x}$ is exactly $k$-sparse.

Theorem

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

Still kind of hard.

Simplest case: $\hat{x}$ is exactly 1-sparse.
Algorithm

Simpler case: \( \hat{x} \) is exactly \( k \)-sparse.

Theorem

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

Still kind of hard.

Simplest case: \( \hat{x} \) is exactly 1-sparse.

Lemma

We can compute a 1-sparse \( \hat{x} \) in \( O(1) \) time.
Algorithm for $k = 1$

Lemma

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

$$\hat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$
Algorithm for $k = 1$

Lemma

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

\[
\hat{x}_i = \begin{cases} 
    a & \text{if } i = t \\
    0 & \text{otherwise}
\end{cases}
\]

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

Lemma

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

\[
\hat{x}_i = \begin{cases} 
    a & \text{if } i = t \\
    0 & \text{otherwise}
\end{cases}
\]

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

  $x_0 = a$
Algorithm for $k = 1$

Lemma

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

\[ \hat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \]

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

- $x_0 = a$
- $x_1 = a\omega^t$
Algorithm for $k = 1$

Lemma

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

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

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

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

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

(Related to OFDM, Prony's method, matrix pencil.)
Algorithm for $k = 1$

**Lemma**

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

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

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

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

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

(Related to OFDM, Prony's method, matrix pencil.)
Algorithm for $k = 1$

**Lemma**

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

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

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

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

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

(Related to OFDM, Prony’s method, matrix pencil.)
Algorithm for general $k$

- Reduce general $k$ to $k = 1$.

![Diagram]
Algorithm for general $k$

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

![Diagram](image)

**Lemma (Partial sparse recovery)**
In $O(k \log n)$ expected time, we can compute an estimate $\hat{x}'$ such that $\|\hat{x} - \hat{x}'\|_{\infty}$ is $\frac{k}{2}$-sparse.
Algorithm for general $k$

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

*Filters*: partition frequencies into $O(k)$ buckets.

1-sparse recovery

$\hat{x}$

$x'$
Algorithm for general $k$

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

![Diagram showing the algorithm](image)

** Lemma (Partial sparse recovery)**
In $O(k \log n)$ expected time, we can compute an estimate $\hat{x}'$ such that $\hat{x} - \hat{x}'$ is $k/2$-sparse.
Algorithm for general $k$

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

### Lemma (Partial sparse recovery)
In $O(k \log n)$ expected time, we can compute an estimate $\hat{x}'$ such that $\hat{x} - \hat{x}'$ is $k/2$-sparse.
Algorithm for general $k$

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

![Diagram showing the algorithm for general $k$]

$x$ → Filters → $\hat{x}'$

$1$-sparse recovery

$O(k)$
Algorithm for general $k$

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

Diagram:
- Input $x$ through filters $O(k)$
- Each filter $1$-sparse recovery
- Output $\hat{x}'$
**Algorithm for general** \(k\)

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

---

**Diagram:**

- \(x\) → Filters → 1-sparse recovery → \(\hat{x}'\)
  - \(O(k)\)
  - 1-sparse recovery

---

**Lemma (Partial sparse recovery):**

In \(O(k \log n)\) expected time, we can compute an estimate \(\hat{x}'\) such that

\[
\|\hat{x} - \hat{x}'\|_1 \leq \frac{k}{2}
\]
Algorithm for general \( k \)

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

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

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

![Diagram](image)

$x$  →  Permute  →  Filters  →  $\hat{x}'$

1-sparse recovery

$O(k)$

1-sparse recovery

1-sparse recovery

1-sparse recovery

Lemma (Partial sparse recovery)

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

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

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

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

Recovers most of $\hat{x}$:

**Lemma (Partial sparse recovery)**

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

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

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

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

Repeat, $k \rightarrow k/2 \rightarrow k/4 \rightarrow \ldots$
**Overall outline**

Partial $k$-sparse recovery

**Lemma (Partial sparse recovery)**

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

Repeat, $k \to k/2 \to k/4 \to \cdots$
Lemma (Partial sparse recovery)

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

Repeat, $k \rightarrow k/2 \rightarrow k/4 \rightarrow \cdots$
Lemma (Partial sparse recovery)

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

Repeat, $k \rightarrow k/2 \rightarrow k/4 \rightarrow \cdots$
Overall outline

Partial $k$-sparse recovery

$x$  
\begin{align*}
\text{Permute} & \quad \text{Filters} \\
& \quad \begin{array}{c}
\rightarrow 1\text{-sparse recovery} \\
\rightarrow 1\text{-sparse recovery} \\
\rightarrow O(k) \\
\rightarrow 1\text{-sparse recovery} \\
\rightarrow 1\text{-sparse recovery}
\end{array} \\
& \rightarrow \hat{x}'
\end{align*}

Lemma (Partial sparse recovery)

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

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

Theorem

We can compute $\hat{x}$ in $O(k \log n)$ expected time.
Lemma (Partial sparse recovery)

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

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

Theorem

We can compute $\hat{x}$ in $O(k \log n)$ expected time.
How can you isolate frequencies?

\[
\begin{align*}
\text{Time} & \quad \text{Frequency} \\
\end{align*}
\]

\[n\text{-dimensional DFT: } O(n \log n)\]
\[x \rightarrow \hat{x}\]
How can you isolate frequencies?

$n$-dimensional DFT: $O(n \log n)$

$x \rightarrow \hat{x}$
How can you isolate frequencies?

\[ \times \rightarrow \hat{x} \]

\[ x \cdot \text{rect} \rightarrow \hat{x} * \text{sinc}. \]

\( n \)-dimensional DFT: \( O(n \log n) \)

\[ x \rightarrow \hat{x} \]

\( n \)-dimensional DFT of first \( k \) terms: \( O(n \log n) \)

\[ x \cdot \text{rect} \rightarrow \hat{x} * \text{sinc}. \]
How can you isolate frequencies?

\[ n \text{-dimensional DFT: } O(n \log n) \]
\[ x \rightarrow \hat{x} \]

\[ n \text{-dimensional DFT of first } k \text{ terms: } O(n \log n) \]
\[ x \cdot \text{rect} \rightarrow \hat{x} \ast \text{sinc.} \]
How can you isolate frequencies?

\[ n \]-dimensional DFT:
\[ O(n \log n) \]
\[ x \rightarrow \hat{x} \]

\[ n \]-dimensional DFT of first \( k \) terms:
\[ O(n \log n) \]
\[ x \cdot \text{rect} \rightarrow \hat{x} \ast \text{sinc} \]

\[ k \]-dimensional DFT of first \( k \) terms:
\[ O(B \log B) \]
\[ \text{alias}(x \cdot \text{rect}) \rightarrow \text{subsample}(\hat{x} \ast \text{sinc}) \]
How can you isolate frequencies?

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

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

$k$-dimensional DFT of first $k$ terms: $O(B \log B)$
alias$(x \cdot \text{rect}) \rightarrow$ subsample$(\hat{x} \ast \text{sinc})$. 
The issue

We want to isolate frequencies.
The issue

We want to isolate frequencies.

The sinc filter “leaks”.
Contamination from other buckets.
The issue

We want to isolate frequencies.

The sinc filter “leaks”.
Contamination from other buckets.

We introduce a better filter.
Algorithm for *exactly sparse* signals

Original signal $x$

Goal $\hat{x}$

Lemma

If $t$ is isolated in its bucket and in the “super-pass” region, the value $b$ we compute for its bucket satisfies $b = \hat{x}_t$.

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

Computed $F \cdot x$

Filtered signal $\hat{F} \ast \hat{x}$
Algorithm for *exactly sparse* signals

$$F \cdot x \text{ aliased to } k \text{ terms}$$

Filtered signal $$\hat{F} \ast \hat{x}$$
Algorithm for *exactly sparse* signals

\[ F \cdot x \text{ aliased to } k \text{ terms} \]

\[ \text{Computed samples of } \hat{F} \ast \hat{x} \]

Lemma

If \( t \) is isolated in its bucket and in the “super-pass” region, the value \( b \) we compute for its bucket satisfies

\[ b = \hat{x}_t. \]

Computing the \( b \) for all \( O(k) \) buckets takes \( O(k \log n) \) time.
Algorithm for exactly sparse signals

$F \cdot x$ aliased to $k$ terms

Computed samples of $\hat{F} \ast \hat{x}$

Lemma
If $t$ is isolated in its bucket and in the "super-pass" region, the value $b$ we compute for its bucket satisfies $b = \hat{x}_t$.

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

\( F \cdot x \) aliased to \( k \) terms

Knowledge about \( \hat{x} \)

Lemma

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

\[ b = \hat{x}_t. \]

Computing the \( b \) for all \( O(k) \) buckets takes \( O(k \log n) \) time.
Algorithm for \textit{exactly sparse} signals

$F \cdot x$ aliased to $k$ terms

Knowledge about $\hat{x}$

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

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

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

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

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

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

**Lemma**

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

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

Computing the $b$ for all $O(k)$ buckets takes $O(k \log n)$ time.
Algorithm for *exactly* sparse signals

**Lemma**

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

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

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

- Time-shift $x$ by one and repeat: $b' = \hat{x}_t \omega^t$.
- Divide to get $b'/b = \omega^t$.
Algorithm for exactly sparse signals

Lemma

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

\[
b = \hat{x}_t.
\]

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

- Time-shift \( x \) by one and repeat: \( b' = \hat{x}_t \omega^t \).
- Divide to get \( b'/b = \omega^t \implies \) can compute \( t \).
Algorithm for \textit{exactly} sparse signals

Lemma

\textit{For most} $t$, \textit{the value} $b$ \textit{we compute for its bucket satisfies}

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

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

\begin{itemize}
  \item Time-shift $x$ by one and repeat: $b' = \hat{x}_t \omega^t$.
  \item Divide to get $b'/b = \omega^t \implies$ can compute $t$.
    \begin{itemize}
      \item Just like our 1-sparse recovery algorithm, $x_1/x_0 = \omega^t$.
    \end{itemize}
\end{itemize}
Algorithm for exactly sparse signals

Lemma

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

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

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

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

Lemma

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

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

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

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

Repeat $k \rightarrow k/2 \rightarrow k/4 \rightarrow \cdots$
Algorithm for exactly sparse signals

Lemma

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

\[
b = \hat{x}_t.
\]

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

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

\[\begin{array}{c}
\text{Permute} \\
\text{Filters} \\
\end{array} \xrightarrow{O(k)} \hat{x}' \]

- Repeat \( k \rightarrow k/2 \rightarrow k/4 \rightarrow \cdots \)
- \( O(k \log n) \) time sparse Fourier transform.
Algorithm for approximately sparse signals
Algorithm for *approximately sparse* signals

- What changes with noise?
Algorithm for *approximately sparse* signals

- What changes with noise?
- Identical architecture:

```
Partial sparse recovery
```

```
\begin{array}{c}
\text{Permute} \\
\text{Filters} \\
\end{array}
```

```
\xrightarrow{O(k)}
```

```
\xrightarrow{\hat{x}'}
```

Just requires robust 1-sparse recovery.

Eric Price (MIT)

A Faster Fourier Transform on Sparse Data

21 / 39
Algorithm for \textit{approximately sparse} signals

- What changes with noise?
- Identical architecture:

```
Partial sparse recovery
```

\begin{itemize}
  \item Just requires robust 1-sparse recovery.
\end{itemize}
Algorithm for *approximately sparse* signals: $k = 1$

**Lemma**

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

$$\left| \hat{x}_t \right| / \| \hat{x} \|_2 \geq 90\%.$$  

Then we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.
Algorithm for *approximately sparse* signals: \( k = 1 \)

**Lemma**

*Suppose \( \hat{x} \) is approximately 1-sparse:*

\[
\frac{|\hat{x}_t|}{\|\hat{x}\|_2} \geq 90\%.
\]

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

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

\( x_1/x_0 = \omega^t \)
Algorithm for \textit{approximately sparse} signals: $k = 1$

\textbf{Lemma}

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

$$|\hat{x}_t|/\|\hat{x}\|_2 \geq 90\%.$$ 

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

- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
Algorithm for \textit{approximately sparse} signals: $k = 1$

Lemma

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

$$\frac{\hat{x}_t}{\|\hat{x}\|_2} \geq 90\%.$$  

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

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

\[x_1/x_0 = \omega^t + \text{noise}\]
Algorithm for \textit{approximately sparse} signals: $k = 1$

Lemma

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

$$|\hat{x}_t|/\|\hat{x}\|_2 \geq 90\%.$$  

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

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

$x_{c_2}/x_0 = \omega^{c_2t} + \text{noise}$
Algorithm for *approximately* sparse signals: \( k = 1 \)

**Lemma**

Suppose \( \hat{x} \) is approximately 1-sparse:

\[
|\hat{x}_t|/\|\hat{x}\|_2 \geq 90\%.
\]

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

- With exact sparsity: \( \log n \) bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose \( \Theta(\log n) \) time shifts \( c \) to recover \( i \).
Algorithm for *approximately sparse* signals: $k = 1$

**Lemma**

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

$$|\hat{x}_t|/\|\hat{x}\|_2 \geq 90\%.$$  

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

- With exact sparsity: $\log n$ bits in a single measurement.
- With noise: only constant number of useful bits.
- Choose $\Theta(\log n)$ time shifts $c$ to recover $i$.
- Error correcting code with efficient recovery $\Rightarrow$ Lemma.
Algorithm for *approximately sparse* signals: general $k$

**Lemma**

If $\hat{x}$ is approximately 1-sparse, we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.
Algorithm for *approximately sparse* signals: general \( k \)

**Lemma**

*If \( \hat{x} \) is approximately 1-sparse, we can recover it with \( O(\log n) \) samples and \( O(\log^2 n) \) time.*

Reduce \( k \)-sparse to 1-sparse on buckets of size \( n/k \), with \( \log n \) overhead per sample.
Algorithm for *approximately sparse* signals: general $k$

**Lemma**

*If $\hat{x}$ is approximately 1-sparse, we can recover it with $O(\log n)$ samples and $O(\log^2 n)$ time.*

Reduce $k$-sparse to 1-sparse on buckets of size $n/k$, with $\log n$ overhead per sample.

**Theorem**

*If $\hat{x}$ is approximately $k$-sparse, we can recover it in $O(k \log(n/k) \log n)$ time.*
Empirical performance

Compare to

- FFTW, the “Fastest Fourier Transform in the West”
- AAFFT, the [GMS05] sparse Fourier transform.
Empirical performance

- Compare to
  - FFTW, the “Fastest Fourier Transform in the West”
  - AAFFT, the [GMS05] sparse Fourier transform.

![Graph showing Run Time vs Signal Sparsity (N=2^{22})]
Empirical performance

- Compare to
  - FFTW, the “Fastest Fourier Transform in the West”
  - AAFFT, the [GMS05] sparse Fourier transform.
Empirical performance

- Compare to
  - FFTW, the “Fastest Fourier Transform in the West”
  - AAFFT, the [GMS05] sparse Fourier transform.

Faster than FFTW for wide range of values.
Recap of Sparse Fourier Transform

Theory:
- The fastest algorithm for Fourier transforms of sparse data.
- The only algorithms faster than FFT for all $k = o(n)$. 

Practice:
- Implementation is faster than FFTW for a wide range of inputs.
- Orders of magnitude faster than previous sparse Fourier transforms.
- Useful in multiple applications.
Recap of Sparse Fourier Transform

**Theory:**
- The fastest algorithm for Fourier transforms of sparse data.
- The only algorithms faster than FFT for all $k = o(n)$.

**Practice:**
- Implementation is faster than FFTW for a wide range of inputs.
- Orders of magnitude faster than previous sparse Fourier transforms.
- Useful in multiple applications.
Talk Outline

1. Sparse Fourier Transform
   - Overview
   - Technical Details

2. Beyond: Sparse Recovery / Compressive Sensing
   - Overview
   - Adaptivity
   - Conclusion
Robustly recover sparse $x$ from linear measurements $y = Ax$. 
Sparse Recovery / Compressive Sensing

Robustly recover sparse $x$ from linear measurements $y = Ax$.
Robustly recover sparse $x$ from linear measurements $y = Ax$. 

Sparse Fourier
Sparse Recovery / Compressive Sensing

Robustly recover sparse $x$ from linear measurements $y = Ax$. 

Sparse Fourier

MRI
Sparse Recovery / Compressive Sensing

Robustly recover sparse $x$ from linear measurements $y = Ax$. 

Sparse Fourier  

MRI  

Single-Pixel Camera
Robustly recover sparse $x$ from linear measurements $y = Ax$. 

Sparse Fourier  

MRI  

Single-Pixel Camera  

Streaming Algorithms  

$A(x + \Delta) = Ax + A\Delta$
Sparse Recovery / Compressive Sensing

Robustly recover sparse $x$ from linear measurements $y = Ax$.

Sparse Fourier

MRI

Single-Pixel Camera

Streaming Algorithms

$A(x + \Delta) = Ax + A\Delta$

Genetic Testing

Eric Price (MIT)
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
- MRI: minimize Fourier sample complexity [GHIKPS13?, IKP13?]
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
- MRI: minimize Fourier sample complexity [GHIKPS13?, IKP13?]
- Camera: use Earth-Mover Distance metric [IP11, GIP10, GIPR11]
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
- MRI: minimize Fourier sample complexity [GHIKPS13?, IKP13?]
- Camera: use Earth-Mover Distance metric [IP11, GIP10, GIPR11]
- Streaming: improved analysis of Count-Sketch [MP13?, PW11, P11]
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
- MRI: minimize Fourier *sample* complexity [GHIKPS13?, IKP13?]
- Camera: use Earth-Mover Distance metric [IP11, GIP10, GIPR11]
- Streaming: improved analysis of Count-Sketch [MP13?, PW11, P11]
- Genetic testing: first asymptotic gain using adaptivity [IPW11, PW13]
My Contributions

- Sparse Fourier: minimize time complexity [HIKP12b, HIKP12a]
- MRI: minimize Fourier *sample* complexity [GHIKPS13?, IKP13?]
- Camera: use Earth-Mover Distance metric [IP11, GIP10, GIPR11]
- Streaming: improved analysis of Count-Sketch [MP13?, PW11, P11]
- Genetic testing: first asymptotic gain using adaptivity [IPW11, PW13]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$. 

Choose $v \in \mathbb{R}^n$, observe $y = \langle v, x \rangle$. Choose another $v$ and repeat as needed.

Output $x'$ satisfying $\|x' - x\|_2 < (1 + \epsilon) \min k$-sparse $x(k) \|x - x(k)\|_2$.

Nonadaptively: $\Theta(k \log (n/k))$ measurements necessary and sufficient. [Candes-Romberg-Tao '06, DIPW '10]

Natural question: does adaptivity help? Studied in [MSW08, JXC08, CHNR08, AWZ08, HCN09, ACD11, ...]

First asymptotic improvement: $O(k \log \log (n/k))$ measurements. [IPW '11]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$.
- Choose $v \in \mathbb{R}^n$, observe $y = \langle v, x \rangle$. 

Nonadaptively: $\Theta(k \log \frac{n}{k})$ measurements necessary and sufficient. [Candes-Romberg-Tao '06, DIPW '10]

Natural question: does adaptivity help?

Studied in [MSW08, JXC08, CHNR08, AWZ08, HCN09, ACD11, ...]

First asymptotic improvement: $O(k \log \log \frac{n}{k})$ measurements. [IPW '11]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$.
- Choose $v \in \mathbb{R}^n$, observe $y = \langle v, x \rangle$.
- Choose another $v$ and repeat as needed.

Nonadaptively: $\Theta(k \log \log (n/k))$ measurements necessary and sufficient. [Candes-Romberg-Tao'06, DIPW'10]

Natural question: does adaptivity help?

First asymptotic improvement: $O(k \log \log (n/k))$ measurements. [IPW'11]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$.
- Choose $\nu \in \mathbb{R}^n$, observe $y = \langle \nu, x \rangle$.
- Choose another $\nu$ and repeat as needed.
- Output $x'$ satisfying
  \[ \| x' - x \|_2 < (1 + \epsilon) \min_{k\text{-sparse } x_{(k)}} \| x - x_{(k)} \|_2 \]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$.
- Choose $v \in \mathbb{R}^n$, observe $y = \langle v, x \rangle$.
- Choose another $v$ and repeat as needed.
- Output $x'$ satisfying

\[
\|x' - x\|_2 < (1 + \epsilon) \min_{k\text{-sparse } x(k)} \|x - x(k)\|_2
\]

- Nonadaptively: $\Theta(k \log(n/k))$ measurements necessary and sufficient. [Candès-Romberg-Tao ’06, DIPW ’10]
Adaptive Sparse Recovery Model

- Unknown approximately $k$-sparse vector $x \in \mathbb{R}^n$.
- Choose $v \in \mathbb{R}^n$, observe $y = \langle v, x \rangle$.
- Choose another $v$ and repeat as needed.
- Output $x'$ satisfying

$$
\|x' - x\|_2 < (1 + \epsilon) \min_{k\text{-sparse } x^{(k)}} \|x - x^{(k)}\|_2
$$

- Nonadaptively: $\Theta(k \log(n/k))$ measurements necessary and sufficient. [Candès-Romberg-Tao ’06, DIPW ’10]
- Natural question: does adaptivity help?
Adaptive Sparse Recovery Model

- Unknown approximately \( k \)-sparse vector \( x \in \mathbb{R}^n \).
- Choose \( v \in \mathbb{R}^n \), observe \( y = \langle v, x \rangle \).
- Choose another \( v \) and repeat as needed.
- Output \( x' \) satisfying
  \[
  \| x' - x \|_2 < (1 + \epsilon) \min_{k\text{-sparse } x_{(k)}} \| x - x_{(k)} \|_2
  \]

- Nonadaptively: \( \Theta(k \log(n/k)) \) measurements necessary and sufficient. [Candès-Romberg-Tao ’06, DIPW ’10]
- Natural question: does adaptivity help?
  - Studied in [MSW08, JXC08, CHNR08, AWZ08, HCN09, ACD11, ...]
Adaptive Sparse Recovery Model

- Unknown approximately \( k \)-sparse vector \( x \in \mathbb{R}^n \).
- Choose \( v \in \mathbb{R}^n \), observe \( y = \langle v, x \rangle \).
- Choose another \( v \) and repeat as needed.
- Output \( x' \) satisfying

\[
\|x' - x\|_2 < (1 + \epsilon) \min_{k\text{-sparse } x_{(k)}} \|x - x_{(k)}\|_2
\]

- Nonadaptively: \( \Theta(k \log(n/k)) \) measurements necessary and sufficient. [Candès-Romberg-Tao ’06, DIPW ’10]
- Natural question: does adaptivity help?
  - Studied in [MSW08, JXC08, CHNR08, AWZ08, HCN09, ACD11, ...]
- First asymptotic improvement: \( O(k \log \log(n/k)) \) measurements. [IPW ’11]
Applications of Adaptivity
Applications of Adaptivity
Applications of Adaptivity
Applications of Adaptivity
Applications of Adaptivity
Applications of Adaptivity
Applications of Adaptivity

A Faster Fourier Transform on Sparse Data
Outline of Algorithm

Theorem

Adaptive $k$-sparse recovery is possible with $O(k \log \log (n/k))$ measurements.
Theorem

Adaptive $k$-sparse recovery is possible with $O(k \log \log (n/k))$ measurements.

Suffices to solve for $k = 1$:

Lemma

Adaptive 1-sparse recovery is possible with $O(\log \log n)$ measurements.
Outline of Algorithm

Theorem

*Adaptive \( k \)-sparse recovery is possible with \( O(k \log \log (n/k)) \) measurements.*

Suffices to solve for \( k = 1 \):

Lemma

*Adaptive 1-sparse recovery is possible with \( O(\log \log n) \) measurements.*
1-sparse recovery: non-adaptive lower bound

**Lemma**

*Adaptive 1-sparse recovery is possible with $O(\log \log n)$ measurements.*
1-sparse recovery: non-adaptive lower bound

Lemma

*Adaptive 1-sparse recovery is possible with $O(\log \log n)$ measurements.*

- Non-adaptive lower bound: why is this hard?
1-sparse recovery: non-adaptive lower bound

**Lemma**

Adaptive 1-sparse recovery is possible with $O(\log \log n)$ measurements.

- Non-adaptive lower bound: why is this hard?
- Hard case: $x$ is random $e_i$ plus Gaussian noise $w$ with $\|w\|_2 \approx 1$. 
1-sparse recovery: non-adaptive lower bound

**Lemma**

*Adaptive 1-sparse recovery is possible with \(O(\log \log n)\) measurements.*

- Non-adaptive lower bound: why is this hard?
- Hard case: \(x\) is random \(e_i\) plus Gaussian noise \(w\) with \(\|w\|_2 \approx 1\).

- Robust recovery must locate \(i\).
1-sparse recovery: non-adaptive lower bound

Lemma

Adaptive 1-sparse recovery is possible with $O(\log \log n)$ measurements.

Non-adaptive lower bound: why is this hard?

Hard case: $x$ is random $e_i$ plus Gaussian noise $w$ with $\|w\|_2 \approx 1$.

Robust recovery must locate $i$.

Observations $\langle v, x \rangle = v_i + \langle v, w \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, for $z \sim N(0, 1)$.
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|^2}{\sqrt{n}} z$, where $z \sim N(0, 1)$
1-sparse recovery: non-adaptive lower bound

- Observe $\langle \mathbf{v}, x \rangle = v_i + \frac{\|\mathbf{v}\|_2}{\sqrt{n}} z$, where $z \sim N(0, 1)$
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|^2}{\sqrt{n}} z$, where $z \sim N(0, 1)$
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, 1)$
1-sparse recovery: non-adaptive lower bound

- Observe \( \langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z \), where \( z \sim N(0, 1) \)
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|^2}{\sqrt{n}} z$, where $z \sim N(0, 1)$
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, 1)$

- Shannon 1948: information capacity

$$I(i, \langle v, x \rangle) \leq \frac{1}{2} \log(1 + \text{SNR})$$

where $\text{SNR}$ denotes the “signal-to-noise ratio,”

$$\text{SNR} = \frac{\mathbb{E}[\text{signal}^2]}{\mathbb{E}[\text{noise}^2]} = \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n} = 1$$
1-sparse recovery: non-adaptive lower bound

- Observe $\langle v, x \rangle = v_i + \frac{\|v\|_2}{\sqrt{n}} z$, where $z \sim N(0, 1)$

- Shannon 1948: information capacity

$$I(i, \langle v, x \rangle) \leq \frac{1}{2} \log(1 + \text{SNR})$$

where $\text{SNR}$ denotes the “signal-to-noise ratio,”

$$\text{SNR} = \frac{\mathbb{E}[	ext{signal}^2]}{\mathbb{E}[	ext{noise}^2]} = \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2 / n} = 1$$

- Finding $i$ needs $\Omega(\log n)$ non-adaptive measurements.
1-sparse recovery: changes in adaptive setting

- Information capacity

\[ I(i, \langle v, x \rangle) \leq \frac{1}{2} \log(1 + \text{SNR}). \]

where \( \text{SNR} \) denotes the “signal-to-noise ratio,”

\[ \text{SNR} = \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}. \]
1-sparse recovery: changes in adaptive setting

- Information capacity

\[ I(i, \langle v, x \rangle) \leq \frac{1}{2} \log(1 + \text{SNR}). \]

where \( \text{SNR} \) denotes the “signal-to-noise ratio,”

\[ \text{SNR} = \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}. \]

- If \( i \) is independent of \( v \), this is \( O(1) \).
1-sparse recovery: changes in adaptive setting

- Information capacity

\[ I(i, \langle v, x \rangle) \leq \frac{1}{2} \log(1 + \text{SNR}). \]

where \( \text{SNR} \) denotes the “signal-to-noise ratio,”

\[ \text{SNR} = \frac{\mathbb{E}[v_i^2]}{\|v\|_2^2/n}. \]

- If \( i \) is independent of \( v \), this is \( O(1) \).
- As we learn about \( i \), we can increase the SNR.
1-sparse recovery: idea

\[ x = e_i + w \]

Signal \[ \rightarrow \] Candidate set

0 bits

\[ SNR = 2 \]

\[ \langle v, x \rangle = v_i + \langle v, w \rangle \]

\[ l(i, \langle v, x \rangle) \leq \log SNR = 1 \]
1-sparse recovery: idea

\[ x = e_i + w \]

Signal  \rightarrow  Candidate set

\begin{align*}
0 \text{ bits} & \quad \text{Candidate set} \\
1 \text{ bit} & \quad \text{Candidate set}
\end{align*}

\[ \mathbf{v} \]

\[ \text{SNR} = 2^2 \]

\[ l(i, \langle \mathbf{v}, \mathbf{x} \rangle) \leq \log \text{SNR} = 2 \]

\[ \langle \mathbf{v}, \mathbf{x} \rangle = v_i + \langle \mathbf{v}, \mathbf{w} \rangle \]
1-sparse recovery: idea

\[ x = e_i + w \]

Signal \[ \rightarrow \] Candidate set

0 bits

1 bit

2 bits

\( SNR = 2^4 \)

\[ l(i, \langle v, x \rangle) \leq \log SNR = 4 \]

\[ \langle v, x \rangle = v_i + \langle v, w \rangle \]
1-sparse recovery: idea

\[ x = e_i + w \]

Signal \[ \rightarrow \] Candidate set

0 bits

1 bit

2 bits

4 bits

\[ \langle v, x \rangle = v_i + \langle v, w \rangle \]

\[ SNR = 2^8 \]

\[ I(i, \langle v, x \rangle) \leq \log SNR = 8 \]
1-sparse recovery: idea

\[ x = e_i + w \]

Signal \[ \rightarrow \] Candidate set

- 0 bits
- 1 bit
- 2 bits
- 4 bits
- 8 bits

\[ \langle v, x \rangle = v_i + \langle v, w \rangle \]

\[ \text{SNR} = 2^{16} \]

\[ I(i, \langle v, x \rangle) \leq \log \text{SNR} = 16 \]
1-sparse recovery

Lemma (IPW11)

Adaptive 1-sparse recovery takes \( O(\log \log n) \) measurements.
1-sparse recovery

Lemma (IPW11, PW13)

Adaptive 1-sparse recovery takes $\Theta(\log \log n)$ measurements.
1-sparse recovery

Lemma (IPW11, PW13)

Adaptive 1-sparse recovery takes $\Theta(\log \log n)$ measurements.
1-sparse recovery

Lemma (IPW11, PW13)

Adaptive 1-sparse recovery takes $\Theta(\log \log n)$ measurements.
Summary

- Sparse Fourier transform
  - Fastest algorithm for Fourier transforms on sparse data
  - Already has applications with substantial improvements
Summary

- **Sparse Fourier transform**
  - Fastest algorithm for Fourier transforms on sparse data
  - Already has applications with substantial improvements

- **Broader sparse recovery theory**
  - Sparse Fourier: minimize time complexity [HIKP12]
  - MRI: minimize Fourier sample complexity [GHIKPS13?, IKP13?]
  - Camera: use Earth-Mover Distance metric [IP11, GIP10, GIPR11]
  - Streaming: improved analysis of Count-Sketch [MP13?, PW11, P11]
  - Genetic testing: first asymptotic gain using adaptivity [IPW11, PW13]
The Future

- Make sparse Fourier applicable to more problems
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity

Tight constants in compressive sensing

Analogous to channel capacity in coding theory.

Lower bound techniques, from information theory, should be strong enough.

Thank You

Eric Price (MIT)

A Faster Fourier Transform on Sparse Data
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity
  - Incorporate stronger notions of structure

Thank You

Eric Price (MIT)

A Faster Fourier Transform on Sparse Data
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity
  - Incorporate stronger notions of structure
- Tight constants in compressive sensing
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity
  - Incorporate stronger notions of structure
- Tight constants in compressive sensing
  - Analogous to channel capacity in coding theory.

Thank You

Eric Price (MIT)
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity
  - Incorporate stronger notions of structure
- Tight constants in compressive sensing
  - Analogous to channel capacity in coding theory.
  - Lower bound techniques, from information theory, should be strong enough.
The Future

- Make sparse Fourier applicable to more problems
  - Better sample complexity
  - Incorporate stronger notions of structure
- Tight constants in compressive sensing
  - Analogous to channel capacity in coding theory.
  - Lower bound techniques, from information theory, should be strong enough.

Thank You