Sample Optimal Fourier Sampling in Any Constant Dimension

Piotr Indyk  Michael Kapralov

MIT
IBM Watson

October 21, 2014
Fourier Transform and Sparsity

Discrete Fourier Transform

Given $x \in \mathbb{C}^n$, compute

$$\hat{x}_i = \sum_{j \in [n]} x_j \omega^{ij},$$

where $\omega$ is the $n$-th root of unity. Assume $n$ is a power of 2.

Fundamental tool:
- Compression (image, audio, video)
- Signal processing
- Data analysis
- Medical imaging (MRI, NMR)
Sparse Fourier Transform

The fast algorithm for DFT is FFT, runs in $O(n \log n)$ time

- improving on FFT runtime in full generality a major open problem
Sparse Fourier Transform

The fast algorithm for DFT is FFT, runs in $O(n \log n)$ time.

- improving on FFT runtime in full generality a major open problem

Most interesting signals are **sparse** (have few nonzero entries) or **approximately sparse** in the Fourier domain.

$k$-sparse = at most $k$ non-zeros

Hassanieh-Indyk-Katabi-Price'12 compute approximate sparse FFT in $O(k \log n \log(n/k))$ time.
Sample complexity

Sample complexity = number of samples accessed in time domain.
In some applications at least as important as runtime

Shi-Andronesi-Hassanieh-Ghazi-Katabi-Adalsteinsson’
ISMRM’13
Sample complexity

Sample complexity = number of samples accessed in time domain. In some applications at least as important as runtime.

Shi-Andronesi-Hassanieh-Ghaziz-Katabi-Adalsteinsson’
ISMRRM’13

Given access to $x \in \mathbb{C}^n$, find $\hat{y}$ such that

$$||\hat{x} - \hat{y}||^2 \leq C \cdot \min_{k\text{-sparse}} \hat{z} ||\hat{x} - \hat{z}||^2$$

Use smallest possible number of samples?
### Uniform bounds (for all):
- Candes-Tao’06
- Rudelson-Vershynin’08
- Candes-Plan’10
- Cheraghchi-Guruswami-Velingker’12

Determined, $\Omega(n)$ runtime
- $O(k \log^3 k \log n)$

### Non-uniform bounds (for each):
- Goldreich-Levin’89
- Kushilevitz-Mansour’91, Mansour’92
- Gilbert-Guha-Indyk-Muthukrishnan-Strauss’02
- Gilbert-Muthukrishnan-Strauss’05
- Hassanieh-Indyk-Katabi-Price’12a
- Hassanieh-Indyk-Katabi-Price’12b
- Indyk-K.-Price’14

Randomized, $O(k \cdot \text{poly}(\log n))$ runtime
- $O(k \log n \cdot (\log \log n)^C)$

### Lower bound:
$\Omega(k \log(n/k))$ for non-adaptive algorithms
- Do-Ba-Indyk-Price-Woodruff’10
Uniform bounds (for all):

Candes-Tao’06
Rudelson-Vershynin’08
Candes-Plan’10
Cheraghchi-Guruswami-Velingker’12

Non-uniform bounds (for each):

Goldreich-Levin’89
Kushilevitz-Mansour’91, Mansour’92
Gilbert-Guha-Indyk-Muthukrishnan-Strauss’02
Gilbert-Muthukrishnan-Strauss’05
Hassanieh-Indyk-Katabi-Price’12a
Hassanieh-Indyk-Katabi-Price’12b
Indyk-K.-Price’14

Deterministic, $\Omega(n)$ runtime

$O(k \log^3 k \log n)$

Randomized, $O(k \cdot \text{poly}(\log n))$ runtime

$O(k \log n \cdot (\log\log n)^C)$

Lower bound: $\Omega(k \log(n/k))$ for non-adaptive algorithms Do-Ba-Indyk-Price-Woodruff’10

**Theorem**

*There exists an algorithm for $\ell_2/\ell_2$ sparse recovery from Fourier measurements using $O(k \log n)$ samples and $O(n \log^3 n)$ runtime.*

Optimal up to constant factors for $k \leq n^{1-\delta}$. First sample-optimal algorithm even if exponential runtime is allowed.
Higher dimensional Fourier transform is needed in some applications

Given $x \in \mathbb{C}^{[n]^d}$, $N = n^d$, compute

$$\hat{x}_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{i^T j} x_i \quad \text{and} \quad x_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{-i^T j} \hat{x}_i$$

where $\omega$ is the $n$-th root of unity, and $n$ is a power of 2.
Previous sample complexity bounds:

- $O(k \log^d N)$ in sublinear time algorithms
  - runtime $k \log^{O(d)} N$, for each

- $O(k \log^4 N)$ for any $d$
  - $\Omega(N)$ time, for all
Previous sample complexity bounds:

- $O(k \log^d N)$ in sublinear time algorithms
  - runtime $k \log^{O(d)} N$, for each
- $O(k \log^4 N)$ for any $d$
  - $\Omega(N)$ time, for all

Our result:

**Theorem**

*There exists an algorithm for $\ell_2/\ell_2$ sparse recovery from Fourier measurements using $O_d(k \log N)$ samples and $O(N \log^3 N)$ runtime.*

Sample-optimal up to constant factors for any constant $d$, first such algorithm.
Previous sample complexity bounds:
- $O(k \log^d N)$ in sublinear time algorithms
  - runtime $k \log^{O(d)} N$, for each
- $O(k \log^4 N)$ for any $d$
  - $\Omega(N)$ time, for all

Our result:

**Theorem**

*There exists an algorithm for $\ell_2/\ell_2$ sparse recovery from Fourier measurements using $O_d(k \log N)$ samples and $O(N \log^3 N)$ runtime.*

Sample-optimal up to constant factors for any constant $d$, first such algorithm.

Also: sublinear time recovery, but with $\log \log^2 n$ loss in sample complexity
( extending Indyk-K.-Price’14 to higher dimensions)
Outline of talk:

1. $\ell_2/\ell_2$ sparse recovery guarantee
2. Summary of techniques for recovery from Fourier measurements
3. Sample-optimal algorithm in $O(N\log^3 N)$ time for $d = 1$
$\ell_2/\ell_2$ sparse recovery guarantees:

$$||\hat{x} - \hat{y}||^2 \leq C \cdot \min_{k\text{-sparse}} \hat{x} ||\hat{x} - \hat{z}||^2$$

$\mu \approx \text{tail noise}/\sqrt{k}$
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[ \| \hat{x} - \hat{y} \|^2 \leq C \cdot \min_{k \text{-sparse}} 2 \| \hat{x} - \hat{z} \|^2 \]

\[ |\hat{x}_1| \geq \ldots \geq |\hat{x}_k| \geq |\hat{x}_{k+1}| \geq |\hat{x}_{k+2}| \geq \ldots \]

Residual error bounded by noise energy \( \text{Err}_k^2(\hat{x}) \)

\[ \text{Err}_k^2(\hat{x}) = \sum_{j=k+1}^{n} |\hat{x}_j|^2 \]

\( \mu \approx \text{tail noise} / \sqrt{k} \)
$\ell_2/\ell_2$ sparse recovery guarantees:

$$||\hat{x} - \hat{y}||^2 \leq C \cdot \text{Err}_k^2(\hat{x})$$

$|\hat{x}_1| \geq \ldots \geq |\hat{x}_k| \geq |\hat{x}_{k+1}| \geq |\hat{x}_{k+2}| \geq \ldots$

Residual error bounded by noise energy $\text{Err}_k^2(\hat{x})$

$$\text{Err}_k^2(\hat{x}) = \sum_{j=k+1}^{n} |\hat{x}_j|^2$$

$\mu \approx \text{tail noise}/\sqrt{k}$
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[
\| \hat{x} - \hat{y} \|^2 \leq C \cdot \text{Err}_k^2(\hat{x})
\]

\[
|\hat{x}_1| \geq \ldots \geq |\hat{x}_k| \geq |\hat{x}_{k+1}| \geq |\hat{x}_{k+2}| \geq \ldots
\]

Residual error bounded by noise energy \( \text{Err}_k^2(\hat{x}) \)

\[
\text{Err}_k^2(\hat{x}) = \sum_{j=k+1}^{n} |\hat{x}_j|^2
\]

\[\mu \approx \text{tail noise} / \sqrt{k}\]
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

\[
\hat{y}_0 \leftarrow 0
\]

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1})$  \[\blacktriangle\] Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

**REFINEMENT**($x$)

**return** dominant Fourier coefficients $\hat{z}$ of $x$ (approximately)

Gilbert-Guha-Indyk-Muthukrishnan-Strauss’02, Akavia-Goldwasser-Safra’03,
Gilbert-Muthukrishnan-Strauss’05, Iwen’10, Akavia’10, Hassanieh-Indyk-Katabi-Price’12a,
Hassanieh-Indyk-Katabi-Price’12b
Recovery from Fourier measurements

Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc:

$$\hat{x} \ast \hat{G}$$
Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points
Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points
Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $\overline{(x \cdot G)} = \hat{x} \ast \hat{G}$
Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $(x \cdot G) = \hat{x} \ast \hat{G}$

\[
(G \cdot x)_f = \sum_{f' \in [n]} \hat{x}_{f'} \hat{G}_{f-f'}
\]
Task: approximate top $k$ coeffs of $\hat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $(\hat{x} \cdot \hat{G}) = \hat{x} \ast \hat{G}$

\[
(G \cdot x)_f = \hat{x}_f + \sum_{f' \in [n], f' \neq f} \hat{x}_{f'} \hat{G}_{f-f'}
\]
**REFINEMENT**($x$)

**return** dominant Fourier coefficients $\hat{Z}$ of $x$ (approximately)

Take $M = C \log n$ independent measurements:

$$y^j \leftarrow (P_{\sigma_j, a_j, q_j} x) \cdot G$$

Estimate each $f \in [n]$ as

$$\hat{w}_f \leftarrow \text{median}\{\tilde{y}_f^1, \ldots, \tilde{y}_f^M\}.$$  

Sample complexity = filter support $\times \log n$
Like hashing heavy hitters into buckets (COUNTSKETCH, COUNTMIN), but buckets leak
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

$\hat{y}_0 \leftarrow 0$

For $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEメント}(x - y_{t-1})$  ▶ Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** \( x \in \mathbb{C}^n \)
\( \hat{y}_0 \leftarrow 0 \)

**For** \( t = 1 \) to \( L \)

- \( \hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1}) \) \( \triangleright \) Takes random samples of \( x - y \)
- Update \( \hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z} \)

In most prior works sampling complexity is

\[
\text{samples per RFINEMENT step } \times \text{ number of iterations}
\]
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** \( x \in \mathbb{C}^n \)

\( \hat{y}_0 \leftarrow 0 \)

**For** \( t = 1 \) to \( L \)

- \( \hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1}) \)  ▶ Takes random samples of \( x - y \)
- Update \( \hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z} \)

In most prior works sampling complexity is

\[
\text{filter support} \times \log n \times \text{number of iterations}
\]
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1})$ \(\triangleright\) Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

In most prior works sampling complexity is

$$\text{filter support} \times \log n \times \text{number of iterations}$$

Lots of work on carefully choosing filters, reducing number of iterations:

- Hassanieh-Indyk-Katabi-Price’12,
- Ghazi-Hassanieh-Indyk-Katabi-Price-Shi’13, Indyk-K.-Price’14

  - still lose $\Omega(\log \log n)$ in sample complexity (number of iterations)
  - lose $\Omega((\log n)^{d-1} \log \log n)$ in higher dimensions
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** \( x \in \mathbb{C}^n \)
\( \hat{y}_0 \leftarrow 0 \)

For \( t = 1 \) to \( L \)

- \( \hat{z} \leftarrow \text{REFINE} \left( x - y_{t-1} \right) \) \quad \text{\( \triangleright \) Takes random samples of \( x - y \)}
- Update \( \hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z} \)

Our sampling complexity is

\[
\text{filter support} \times \log n \times \text{number of iterations}
\]
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1})$  \(\triangleright \text{Takes random samples of } x - y\)
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

Our sampling complexity is

filter support $\times \log n \times \rightarrow \text{number of iterations}$
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1})$ \(\triangleright \text{Takes random samples of } x - y\)
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

Our sampling complexity is

samples per Refined step $\rightarrow$ number of iterations
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** \( x \in \mathbb{C}^n \)

\( \hat{y}_0 \leftarrow 0 \)

**For** \( t = 1 \) to \( L \)

- \( \hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1}) \) \( \triangleright \) Takes random samples of \( x - y \)
- Update \( \hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z} \)

Our sampling complexity is

\[
\text{samples per RFINEMENT step } \times \text{number of iterations}
\]

Do not use fresh randomness in each iteration! In general challenging:

- only one paper Bayati-Montanari’11 gives provable guarantees, with Gaussians
Iterative refinement

Many algorithms use the iterative refinement scheme:

**Input:** $x \in \mathbb{C}^n$

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{REFINEMENT}(x - y_{t-1})$ \( \triangleright \) *Takes random samples* of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

Our sampling complexity is

```
samples per R\text{EFINEMENT} step $\times$ number of iterations
```

Do not use fresh randomness in each iteration! In general challenging: only one paper Bayati-Montanari’11 gives provable guarantees, with Gaussians

Can use very simple filters! (Essentially) boxcar filter
Sample-optimal algorithm

\[ G \leftarrow B \ast B \ast B \]

Let \( y^m \leftarrow (P_m x) \cdot G \)

\[ m = 0, \ldots, M = C \log n \]

\[ \hat{z}_0 \leftarrow 0 \]

For \( t = 1, \ldots, T = O(\log n) \):

For \( f \in [n] \):

\[ \hat{w}_f \leftarrow \text{median}\{ \tilde{y}^1_f, \ldots, \tilde{y}^M_f \} \]

If \( |\hat{w}_f| < 2^{T-t} \mu / 3 \) then

\[ \hat{w}_f \leftarrow 0 \]

End

\[ \hat{z}_{t+1} = \hat{z}_t + \hat{w} \]

\[ y^m \leftarrow y^m - (P_m w) \cdot G \]

for \( m = 1, \ldots, M \)

End

▷ Take samples of \( x \)

▷ Loop over thresholds

▷ Estimate, prune small elements

▷ Update samples
\( G \leftarrow B \ast B \ast B \)

Let \( y^m \leftarrow (P_m x) \cdot G \)

\( m = 0, \ldots, M = C \log n \)

\( \hat{z}_0 \leftarrow 0 \)

For \( t = 1, \ldots, T = O(\log n) \):

For \( f \in [n] \):

\( \hat{w}_f \leftarrow \text{median}\{ \tilde{y}_f^1, \ldots, \tilde{y}_f^M \} \)

If \( |\hat{w}_f| < 2^{T-t} \mu/3 \) then

\( \hat{w}_f \leftarrow 0 \)

End

\( \hat{z}_{t+1} = \hat{z}_t + \hat{w} \)

\( y^m \leftarrow y^m - (P_m w) \cdot G \)

for \( m = 1, \ldots, M \)

End
Sample-optimal algorithm

\[ G \leftarrow B \ast B \ast B \]

Let \( y^m \leftarrow (P_m x) \cdot G \)
\( m = 0, \ldots, M = C \log n \)

\( \hat{z}_0 \leftarrow 0 \)

For \( t = 1, \ldots, T = O(\log n) \):

For \( f \in [n] \):

\[ \hat{w}_f \leftarrow \text{median}\{\tilde{y}_f^1, \ldots, \tilde{y}_f^M\} \]

If \( |\hat{w}_f| < 2^{T-t} \mu / 3 \) then

\[ \hat{w}_f \leftarrow 0 \]

End

\( \hat{z}_{t+1} = \hat{z}_t + \hat{w} \)

\( y^m \leftarrow y^m - (P_m w) \cdot G \)

for \( m = 1, \ldots, M \)

End
$G \leftarrow B \ast B \ast B$

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\hat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\hat{w}_f \leftarrow \text{median}\{\tilde{y}_1^f, \ldots, \tilde{y}_M^f\}$

If $|\hat{w}_f| < 2^{T-t} \mu / 3$ then

$\hat{w}_f \leftarrow 0$

End

$\hat{z}_{t+1} = \hat{z}_t + \hat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End
G ← B * B * B
Let \( y^m \leftarrow (P_m x) \cdot G \)
\( m = 0, \ldots, M = C \log n \)

\[ \hat{z}_0 \leftarrow 0 \]
For \( t = 1, \ldots, T = O(\log n) \):
For \( f \in [n] \):
\[ \hat{w}_f \leftarrow \text{median} \{ \tilde{y}_f^1, \ldots, \tilde{y}_f^M \} \]
If \( |\hat{w}_f| < 2^{T-t}\mu/3 \) then
\[ \hat{w}_f \leftarrow 0 \]
End

\[ \hat{z}_{t+1} = \hat{z}_t + \hat{w} \]
\[ y^m \leftarrow y^m - (P_m w) \cdot G \]
For \( m = 1, \ldots, M \)
End
$G \leftarrow B \ast B \ast B$

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\hat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\hat{w}_f \leftarrow \text{median}\{\tilde{y}_f^1, \ldots, \tilde{y}_f^M\}$

If $|\hat{w}_f| < 2^{T-t}\mu/3$ then

$\hat{w}_f \leftarrow 0$

End

$\hat{z}_{t+1} = \hat{z}_t + \hat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End
Sample-optimal algorithm

\( G \leftarrow B \ast B \ast B \)

Let \( y^m \leftarrow (P_m x) \cdot G \)

\( m = 0, \ldots, M = C \log n \)

\( \hat{z}_0 \leftarrow 0 \)

For \( t = 1, \ldots, T = O(\log n) \):

For \( f \in [n] \):

\( \hat{w}_f \leftarrow \text{median}\left\{ \tilde{y}_f^1, \ldots, \tilde{y}_f^M \right\} \)

If \( |\hat{w}_f| < 2^{T-t} \mu / 3 \) then

\( \hat{w}_f \leftarrow 0 \)

End

\( \hat{z}_{t+1} = \hat{z}_t + \hat{w} \)

\( y^m \leftarrow y^m - (P_m w) \cdot G \)

For \( m = 1, \ldots, M \)

End

Main challenge: lack of fresh randomness. Why does median work?
Let \( S \) denote the set of heavy hitters:

\[
S = \{ i \in [n] : |\hat{x}_i| > \mu \}.
\]

There cannot be too many of them: \(|S| = O(k)\)
Let $S$ denote the set of heavy hitters:

$$S = \{ i \in [n] : |\hat{x}_i| > \mu \}.$$

There cannot be too many of them: $|S| = O(k)$

Main invariant: never modify $\hat{x}$ outside of $S$

Prove: samples taken have error-correcting properties wrt set $S$ of head elements

Set of head elements does not change, only their values
Problem: recover support of a random \( k \)-sparse signal from Fourier measurements.

Parameters: \( n = 2^{15}, \, k = 10, 20, \ldots, 100 \)

Filter: boxcar filter with support \( k + 1 \)
Comparison to $\ell_1$-minimization (SPGL1)

$O(k \log^3 k \log n)$ sample complexity, requires LP solve

Within a factor of 2 of $\ell_1$ minimization
Open questions

$O(k \log n)$ samples in $O(k \log^{O(1)} n)$ time?

Thank you!