# Rapidly Computing Sparse Chebyshev and Legendre Coefficient Expansions via SFTs

Mark Iwen

Michigan State University

October 18, 2014

Work with **Janice (Xianfeng) Hu** – Graduating in May!

# Computing Sparse Chebyshev Expansions

## The Problem: Rapidly Recover $\Omega$ and $a_\omega$'s

$$f(x) = \sum_{\omega \in \Omega} a_\omega T_\omega(x), \quad \Omega \subset [N], \quad |\Omega| = k \ll N$$

- $T_\omega(x)$ is the Chebyshev polynomial of degree $\omega \in [N]$

$$T_\omega(x) := \cos\left(\omega \arccos(x)\right)$$

- Choosing different samples yields an immediate solution!
  - Sample according to $\cos\left(\frac{2\pi j}{N}\right)$ for $j \in [N]$ and we get

$$f\left(\cos\left(\frac{2\pi j}{N}\right)\right) = \sum_{\omega \in \Omega} a_\omega T_\omega\left(\cos\left(\frac{2\pi j}{N}\right)\right) = \sum_{\omega \in \Omega} a_\omega \cos\left(\frac{2\pi j \omega}{N}\right)$$
$$= \sum_{\omega \in \Omega} \frac{a_\omega}{2}\left(\mathbb{e}^{\frac{2\pi \mathbb{i} j \omega}{N}} - \mathbb{e}^{\frac{-2\pi \mathbb{i} j \omega}{N}}\right).$$

  - So ... sample according to $\cos\left(\frac{2\pi j}{N}\right)$ and use an SFT!

# Sparse Fourier Transforms (SFTs)

## SFTs: Rapidly Recover $\Omega$ and $a_\omega$'s for trigonometric polynomials

$$f(x) = \sum_{\omega \in \Omega} a_\omega \mathbb{e}^{2\pi \mathbb{i} \omega x}, \quad \Omega \subset [N], \quad |\Omega| = k \ll N$$

- Work by Mansour, Gilbert, Guha, Muthukrishnan, Strauss, Hassanieh, Indyk, Katabi, Price, Lawlor, Wang, Christlieb, ...
- Fastest variants recover signals w.h.p. in $\mathcal{O}(k \cdot \log^c N)$-time
- Several variants have theoretical error guarantees that mirror compressive sensing guarantees.
- One can recover a sparse vector $\vec{a}_{\rm s}$ in $\mathcal{O}\left(\frac{k \cdot \log^5 N}{\epsilon \cdot \log\log N}\right)$-time s.t. w.h.p.

$$\left\|\vec{a} - \vec{a}_{\rm s}\right\|_2 \leq \left\|\vec{a} - \vec{a}_k^{\rm opt}\right\|_2 + \frac{\epsilon \cdot \left\|\vec{a} - \vec{a}_{(k/\epsilon)}^{\rm opt}\right\|_1}{\sqrt{k}}.$$

# A Sublinear-time Sparse Chebyshev Algorithm

## The Problem: Rapidly Recover $\Omega$ and $a_\omega$'s

$$f(x) = \sum_{\omega \in \Omega} a_\omega T_\omega(x), \quad \Omega \subset [N], \quad |\Omega| = k \ll N$$

## A Solution

- Run the SFT of your choice on

$$g(x) := f(\cos x) = \sum_{\omega \in \Omega} \frac{a_\omega}{2} \left( \mathrm{e}^{\frac{2\pi \mathrm{i} j \omega}{N}} - \mathrm{e}^{\frac{-2\pi \mathrm{i} j \omega}{N}} \right).$$

- Learn $\left(\omega, \frac{a_{|\omega|}}{2}\right)$ for all $\omega \in \Omega \cup -\Omega$.
- Discard negative frequencies, and double each Fourier coefficient for positive frequencies in order to recover $f(x)$.

# What About Other Polynomial Expansions?

- Legendre polynomials are another natural choice that arise in many applications (spherical harmonics).
- Recursive Definition

$$
\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
&\vdots \\
P_{n+1}(x) &= \frac{2n+1}{n+1} \cdot x \cdot P_n(x) - \frac{n}{n+1} \cdot P_{n-1}(x)
\end{aligned}
$$

- Orthogonal on $[-1, 1]$
- Can we sample with respect to a different function and then just apply an SFT again?

# Related Work: Sparse Legendre Expansions

## The Problem: Recover $\Omega$ and $a_\omega$'s

$$f(x) = \sum_{\omega \in \Omega} a_\omega P_\omega(x), \quad \Omega \subset [N], \quad |\Omega| = k \ll N$$

- Prony-Like Approaches: Potts, Tasche, ...
  - $\mathcal{O}(k)$ unequally-spaced deterministic samples near zero.
  - Uses the SVD/QR of a Hankel/Toeplitz matrix: $\mathcal{O}(k^3)$-time.
  - Show numerical robustness to noise.
- Compressive Sensing Approaches: Rauhut, Ward, ...
  - $\mathcal{O}(k \cdot \log^4 N)$ random samples (i.i.d. from Chebyshev measure).
  - Use Basis Pursuit, OMP, ..., so $\Omega(N)$-time.
  - Show theoretical and numerical robustness to noise.
- Today we will discuss how SFTs can provide robust recovery with $\mathcal{O}(k \cdot \log^c N)$-samples/time.

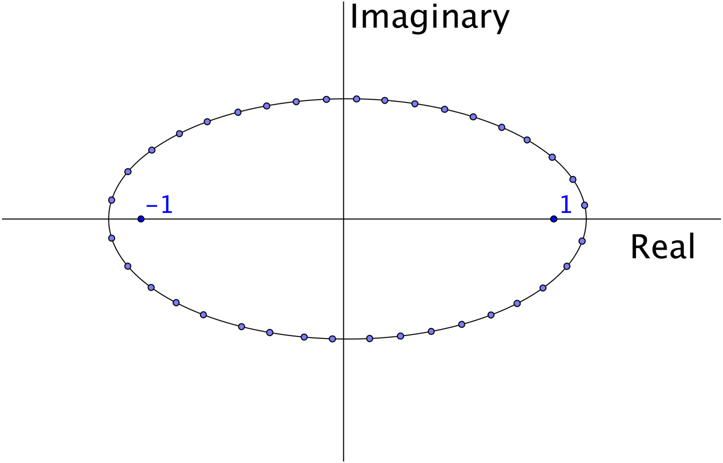
# Related Work: Fast Methods for Standard Legendre

## The Standard Problem: Recover $a_0, a_2, \ldots, a_{N-1}$

$$f(x) = \sum_{n=0}^{N-1} a_n P_n(x)$$

- Want to solve for Legendre Coefficients in $o(N^2)$ time.
- Alpert, Rokhlin, Potts, Steidl, Tasche, Iserles, ...
- Iserles reduces problem to FFT calculation + postprocessing
  - Sample $f$ at $N$ complex values
  - Take the FFT of the $N$ samples
  - Apply a fast linear transform to the FFT result to get $a_0, a_2, \ldots, a_{N-1}$
  - $\mathcal{O}(N \log N)$-time method
- Can an SFT replace the FFT above in sparse setting?
- Is sparsity preserved by this process?

# Isreles' Method: Evaluate on Ellipse in Complex Plane




- We take the FFT of the function $f$ evaluated at points on an ellipse in the complex plane. Get $\kappa_0, \ldots, \kappa_{N-1} \in \mathbb{C}$.

# Isreles' Method: Fast Post-processing

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{N-3} \\ a_{N-2} \\ a_{N-1} \end{pmatrix} \approx \begin{pmatrix} * & \cdots & * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & \cdots & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & \cdots & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & \cdots & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & \cdots & * & 0 \\ & & & & \vdots & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \end{pmatrix} \begin{pmatrix} \kappa_0 \\ \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_{N-3} \\ \kappa_{N-2} \\ \kappa_{N-1} \end{pmatrix}$$

- Post processing is equivalent to multiplying $\vec{\kappa}$ by an upper triangular banded matrix.
- Clearly, $\vec{\kappa}$ sparse $\implies$ $\vec{a}$ is sparse + small errors.
- Does $\vec{a}$ sparse $\implies$ $\vec{\kappa}$ compressible? How compressible?
- If $\vec{\kappa} \in \mathbb{C}^N$ is sparse we can use an SFT.

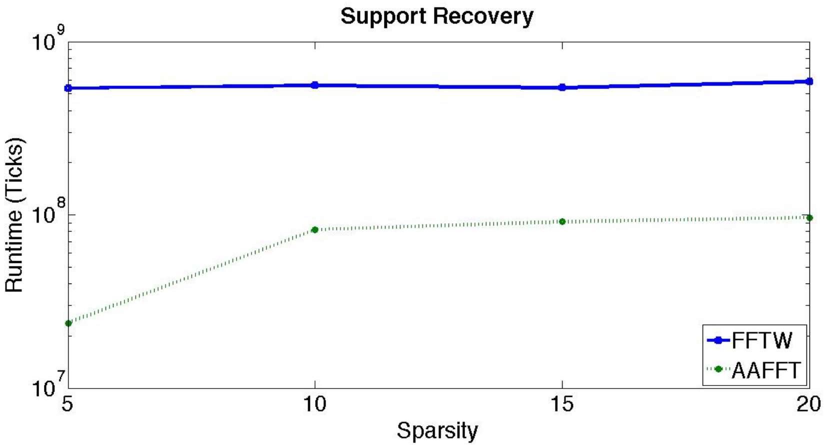
# It Works Pretty Well (so far...)

**The Problem: Recover $\Omega$ and $a_\omega$'s**

$$f(x) = \sum_{\omega \in \Omega} a_\omega P_\omega(x), \quad \Omega \subset [N], \quad |\Omega| = k \ll N$$

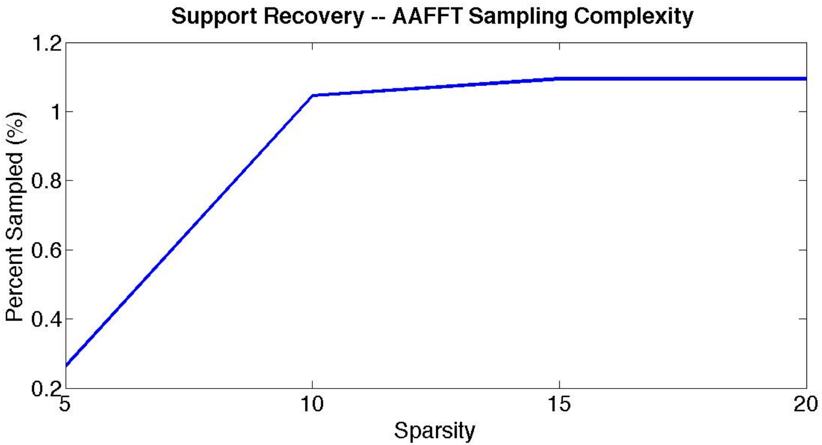
- $\vec{\kappa}$ does indeed appear to be compressible.
- So, we can approximate $\vec{\kappa}$ with an SFT.
- Once we have $\vec{\kappa}$ we can use Isreles' Post-processing method, **OR**
- We can find the support of $\vec{\kappa}$, and then use Rauhut and Ward's RIP-based methods to finish estimating it.
- End result: Janice and I have a $\mathcal{O}(k \cdot \log^c N)$-time method for recovering Legendre-sparse $f$.
- Preprint in progress...

# Runtime: Support Recovery with Probability $\geq 0.7$




- $N = 2^{21} = 2,097,152$

# Sampling: Support Recovery with Probability $\geq 0.7$




- $N = 2^{21} = 2,097,152$

# Thank You!

Questions?