Sublinear Time Spectral Density Estimation

Christopher Musco

New York University, Tandon School of Engineering

COLLABORATORS

Aditya Krishnan
(JHU)

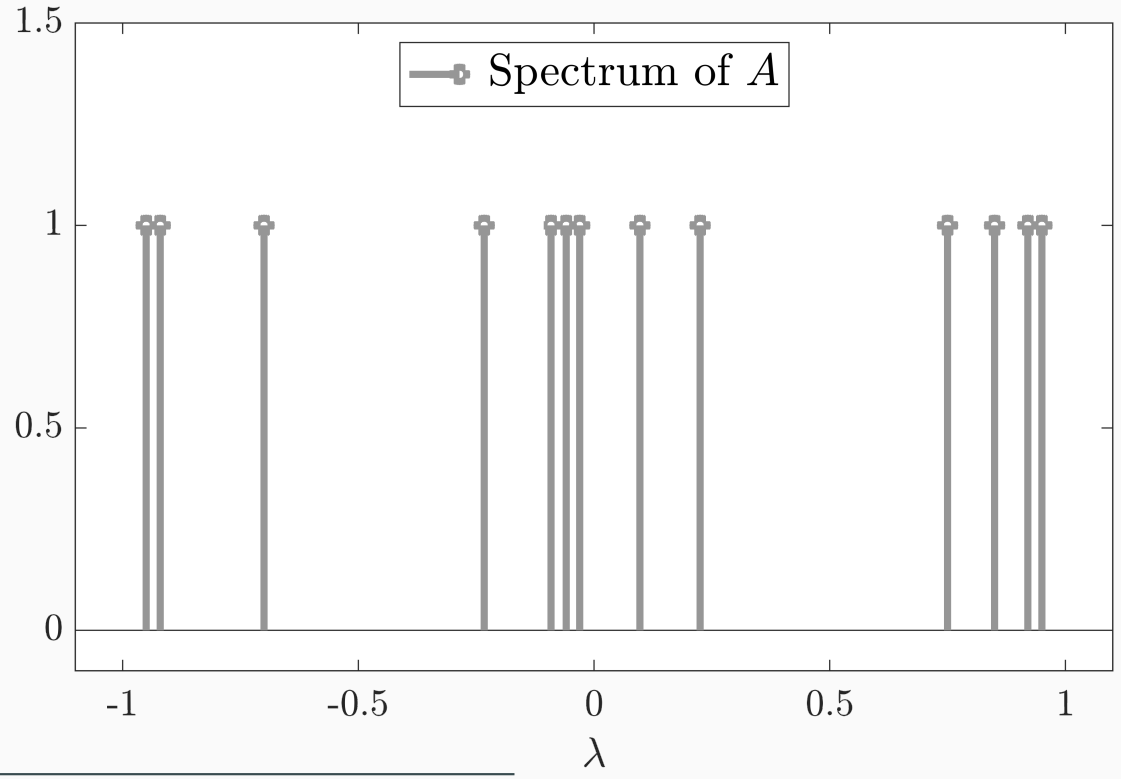
Vladimir Braverman
(JHU)

Paper available on arXiv: https://arxiv.org/abs/2104.03461.

To appear in Symposium on Theory of Computing (STOC 2022).

Basic problem in linear algebra:

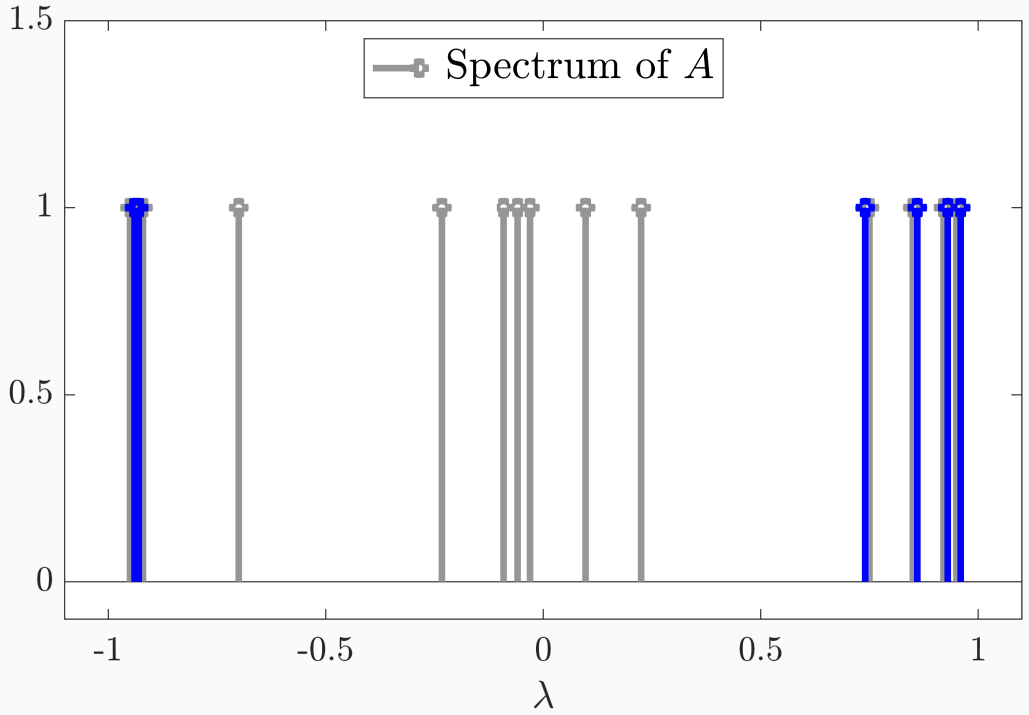
- Given a diagonalizable[1] $n \times n$ matrix $\mathbf{A}$ with real eigenvalues $\lambda_1, \ldots, \lambda_n$.
- Goal is to approximate this spectrum in $\ll O(n^3)$ time.



[1] Let's assume symmetric today, with eigenvalues in $[-1, 1]$.

SPECTRAL DENSITY ESTIMATION

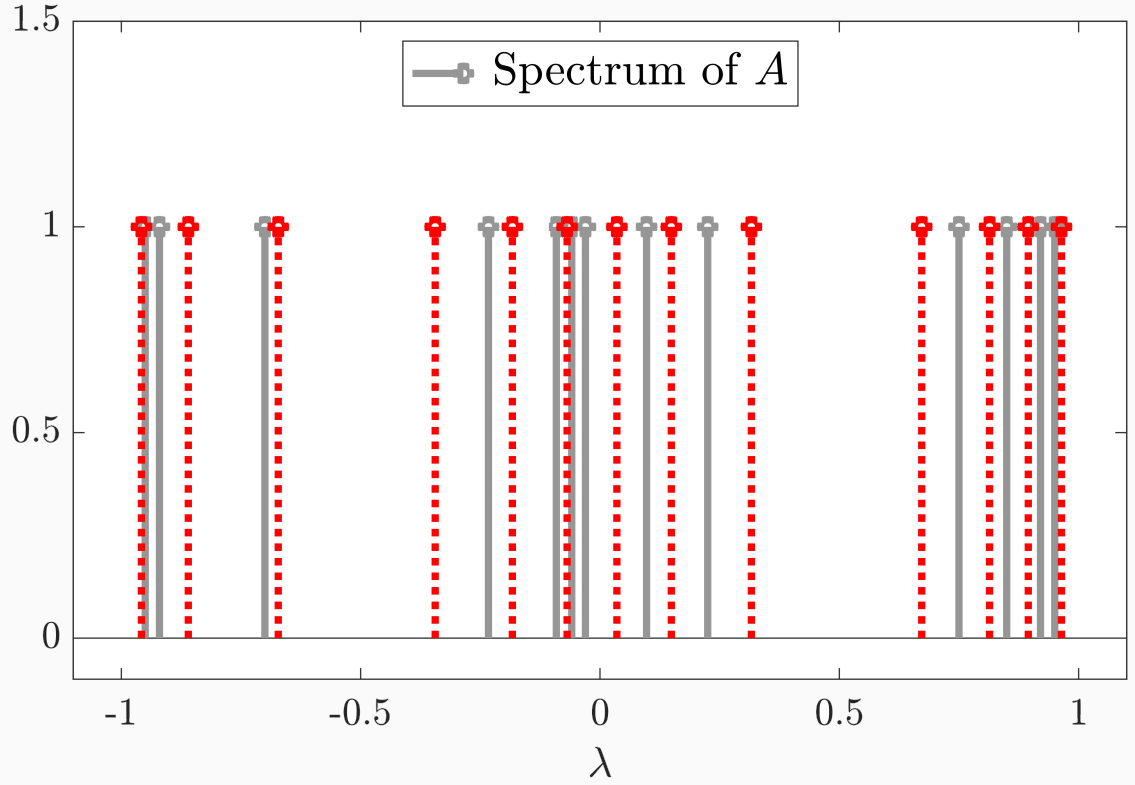
Possible approach: Compute a few outlying eigenpairs of A using an iterative method, like Lanczos iteration.



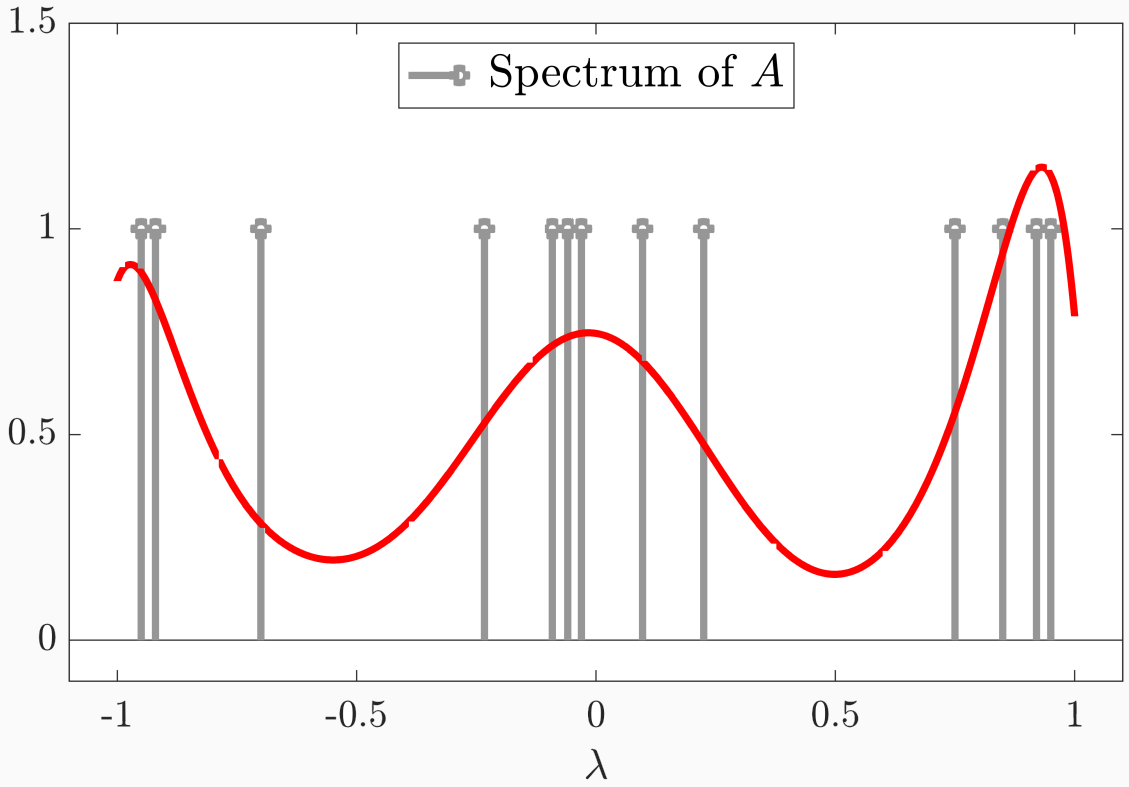
Access **A** via a small number of matrix-vector multiplications, which can be implemented in $O(n^2)$ time or faster.

Can also be applied to implicit matrices.

We want to capture information about the whole spectrum.



Would also be happy with a “smooth” approximation to the eigenvalue distribution



Easily discretized if approximate eigenvalues are desired.

SPECTRAL DENSITY ESTIMATION

View spectrum as a probability density. If $\mathbf{A}$ has eigenvalues $\lambda_1, \ldots, \lambda_n$,

$$\text{Spectral density:} \qquad s(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \lambda_i).$$

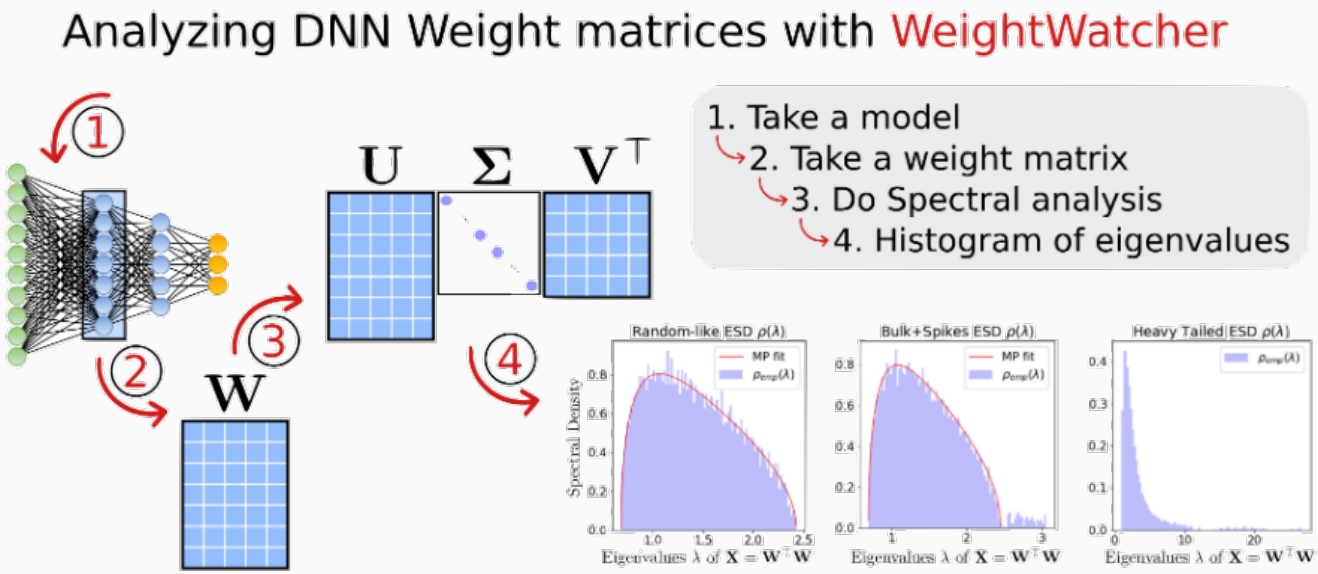
Goal: Find density q which is close to s is some statistical distance.

EXAMPLE APPLICATIONS

- Computational physics and chemistry. [Weiße, Wellein, Alvermann, Fehske 2006]
- Subroutine used to initialize parallel eigensolvers, like the FEAST eigensolver [Polizzi, 2009].
- Approximate spectral sums: $\sum_{i=1}^{n} f(\lambda_i) \approx \sum_{i=1}^{n} f(\tilde{\lambda}_i)$:
  - Matrix norms
  - Log determinant: $f(x) = log(x)$.
  - Estrada index: $f(x) = \mathsf{exp}(x)$.
  - Number of triangles in a graph: $f(x) = x^3$.

EXAMPLE APPLICATIONS

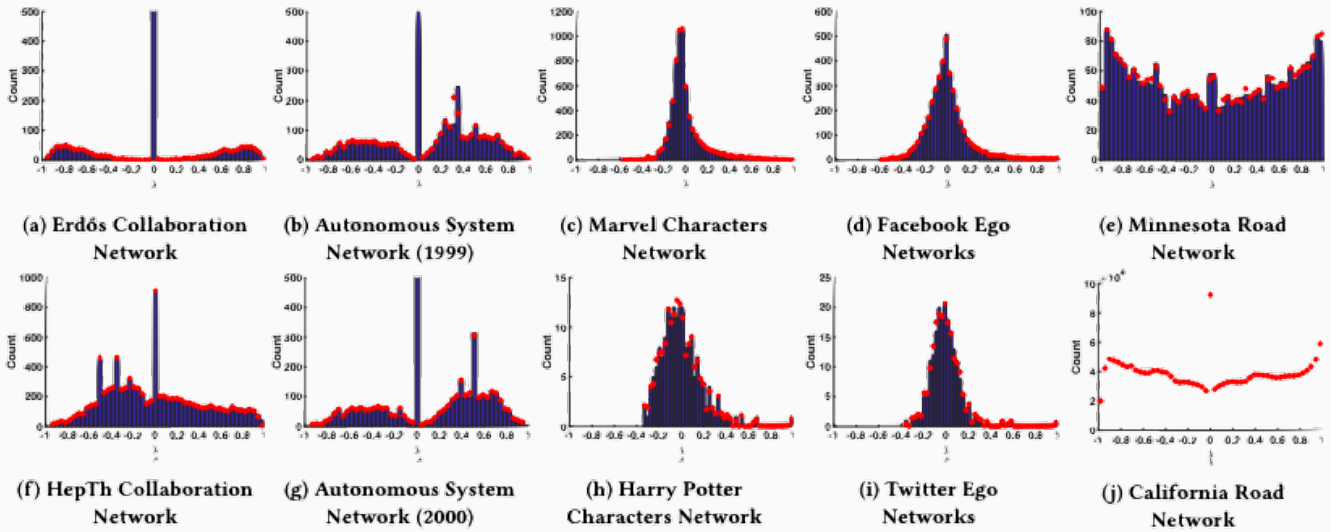
Analyzing the spectra of weight matrices and Hessian matrices in deep learning. Understanding generalization, improving convergence, optimization methods, etc.



Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data, [Martin, Peng, Mahoney, Nature Comm. 2021].

EXAMPLE APPLICATIONS

Analyzing graph structure in network science. E.g. the adjacency matrix of a social network graph.



Network Density of States, [Dong, Benson, Bindel, KDD 2019].

EXISTING METHODS

There has a been a lot of work on this problem, and many methods proposed to solve it.

- Kernel Polynomial Method (KPM)
- Lanczos Spectroscopic Method (SLQ)
- Delta-Gauss-Legendre quadrature
- Lanczos Method for CDF
- Explicit Moment Matching (MM)

See [Lin, Saad, Yang, 2014] for a good overview.

MAIN RESULT

Emerging result: Several of these methods (KPM, SLQ, MM) can provably compute an ϵ-approximate spectral density using just $O\left(\frac{1}{\epsilon}\right)$ matrix vector multiplications with $\mathbf{A}$.

Worst case $O(n^2/\epsilon)$ time for a dense $n \times n$ matrix.

[Chen, Trogdon, Ubaru. ICML 2021] proves a result for the Stochastic Lanczos Quadrature Method.

We focus on the kernel polynomail and moment matching methods.

WHAT DO WE MEAN BY APPROXIMATE?

View spectrum as a probability density. If $\mathbf{A}$ has eigenvalues $\lambda_1, \ldots, \lambda_n$,

$$\text{Spectral density:} \qquad s(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - \lambda_i).$$

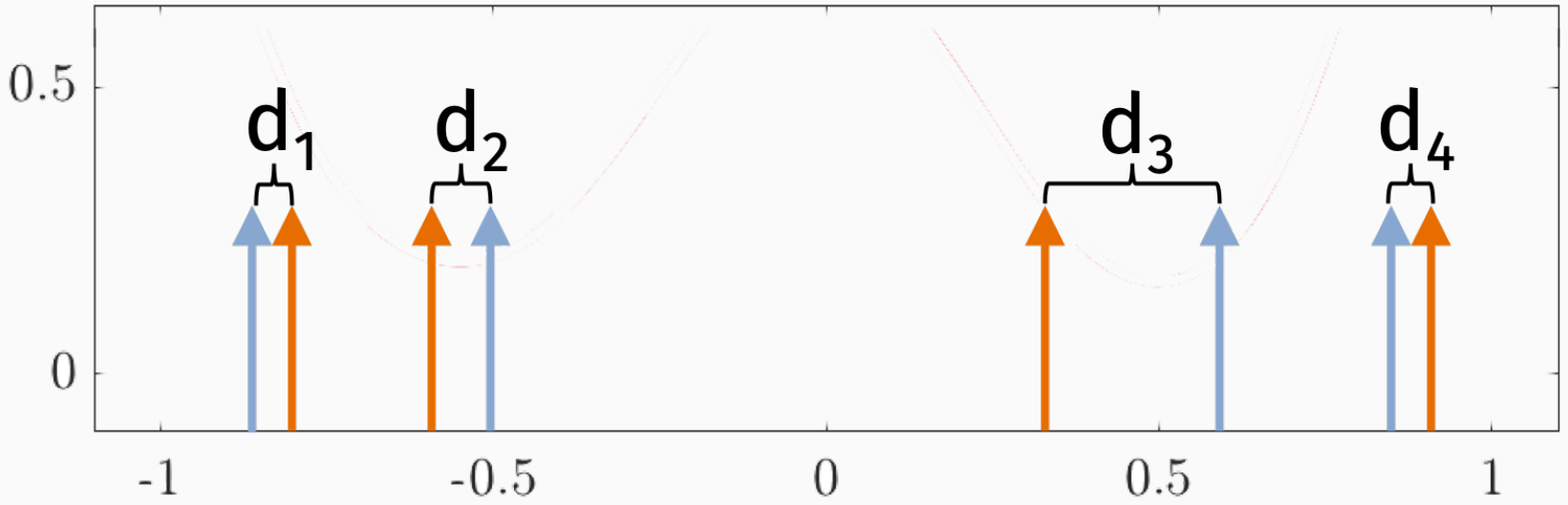
Goal: Find density q which is close to s is some statistical distance.

Natural choice: Wasserstein-1 distance $W_1(s, q)$. Aka "earth mover's distance".

Goal is to find q with $W_1(s, q) \leq \epsilon$.

WASSERSTEIN DISTANCE

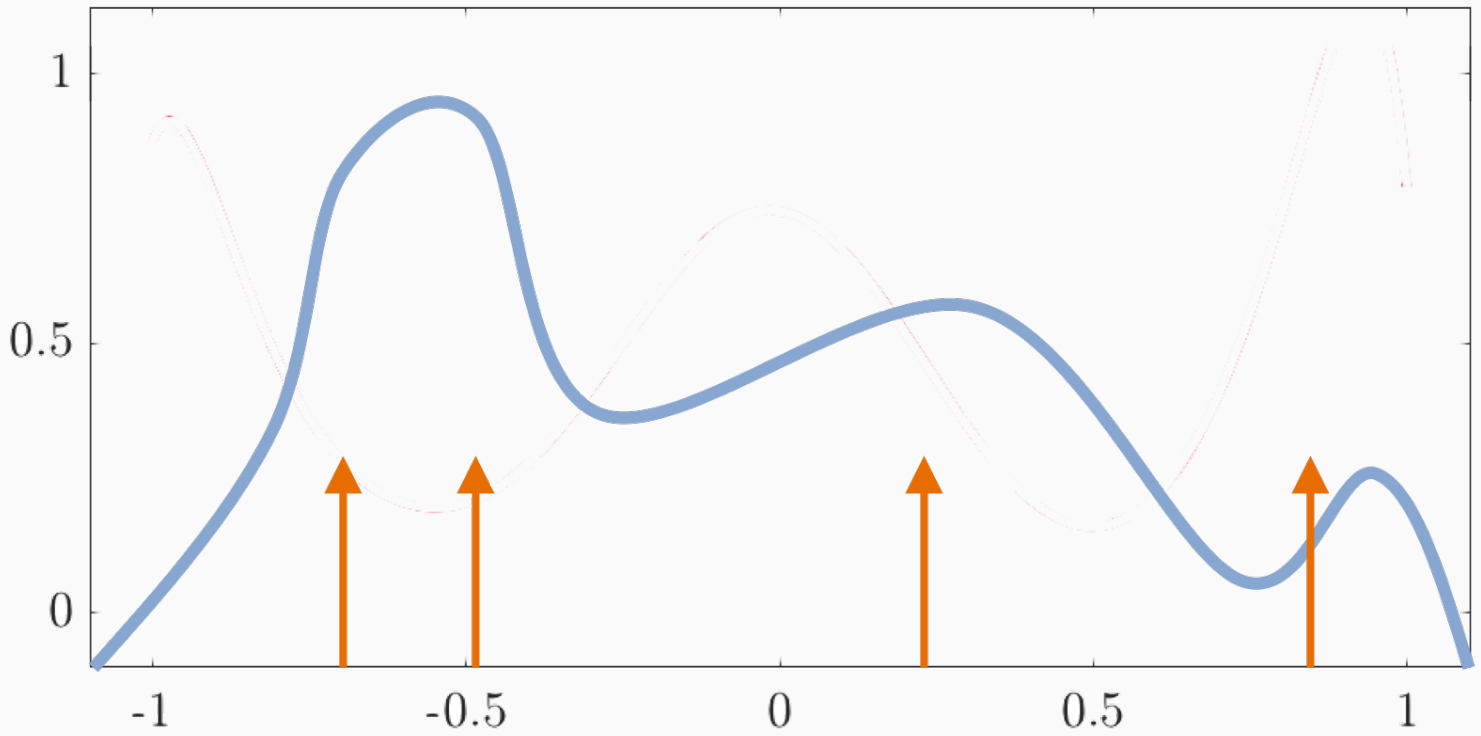
Compute cost of optimal transport plan for moving one distribution to another. E.g. for two point-mass distributions:



$$W_1(s, q) = \frac{1}{4}(d_1 + d_2 + d_3 + d_4).$$

For spectral densities s, q with eigenvalues $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{\lambda}}$,
$W_1(s, q) = \frac{1}{n}\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|_1$.

Nice property: can also be used compare continuous and point mass distributions.



Assume distribution are supported on $[-1, 1]$ and let f be a 1-Lipschitz function from $[-1, 1] \to \mathbb{R}$.

I.e. $|f(x) - f(y)| < |x - y|$ for all x, y.

Dual characterization:

$$W_1(s, q) = \max_{\text{1-Lipschitz } f} \langle f, s - q \rangle$$

where $\langle a, b \rangle = \int_{-1}^{1} a(x)b(x)dx$.

This characterization immediately suggests an approach to obtaining an accurate SDE:

- Let $\mathcal{P}$ be a projection operator onto the first k Chebyshev polynomials.
- Return $q = \mathcal{P}^T s$

$$\begin{aligned} W_1(s, q) = \max_f \langle f, s - q \rangle &= \max_f \langle f, (\mathcal{I} - \mathcal{P}^T) s \rangle \\ &= \max_f \langle (\mathcal{I} - \mathcal{P}) f, s \rangle \\ &= \max_f \langle f - \mathcal{P} f, s \rangle \end{aligned}$$

Since f is 1-Lipschitz, standard results tell use that $f - \mathcal{P}f$ is small. In particular, we have:

$$\|f - \mathcal{P}f\|_\infty \leq O\left(\frac{1}{k}\right).$$

If $q = \mathcal{P}^T s$ where $\mathcal{P}$ projects onto the first $O(1/\epsilon)$ **Chebyshev polynomails** then we have:

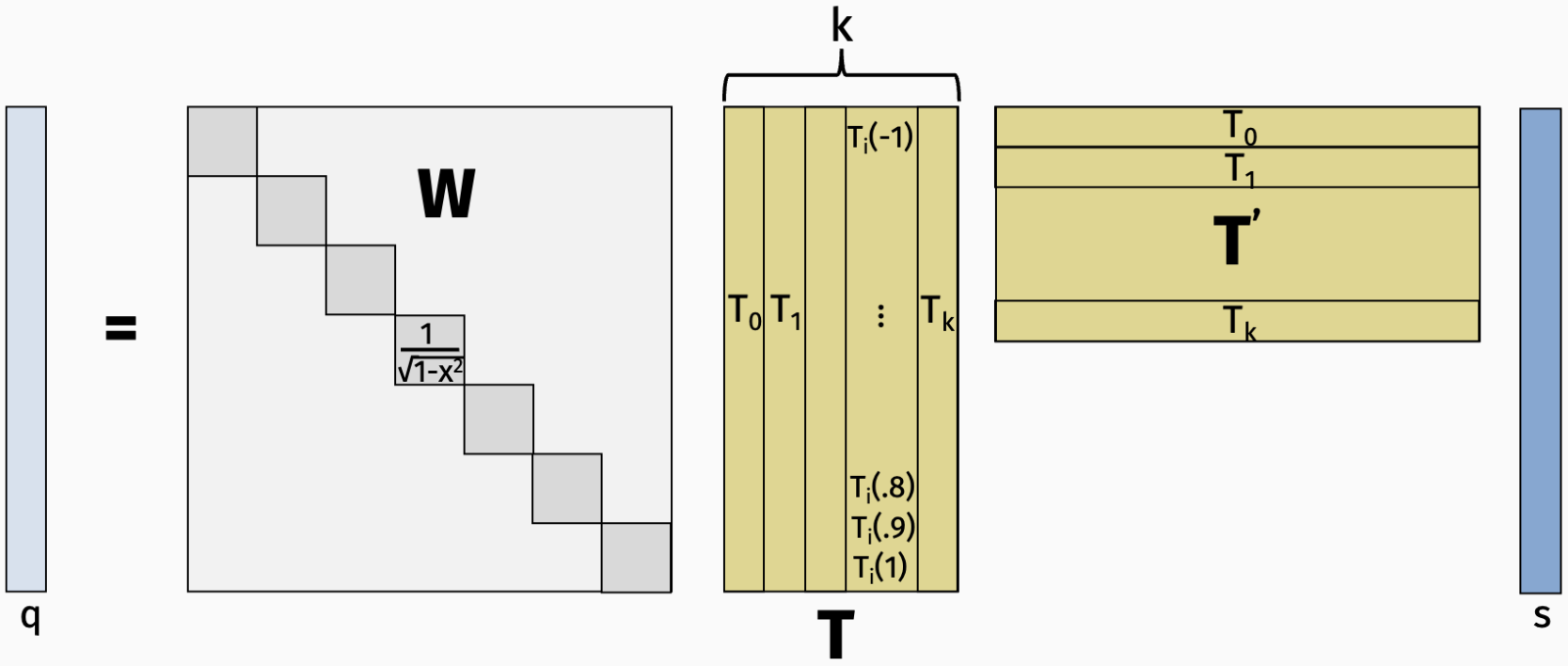
$$W_1(s, q) = \max_f \langle f - \mathcal{P}f, s \rangle \leq \|f - \mathcal{P}f\|_\infty \|s\|_1 \leq \epsilon.$$

That's it!

Two items remain to resolve:

1. How to ensure q is a positive density? We use a Jackson damped Chebyshev expansion instead.
2. How to actually compute q? Let's discuss this next.

POLYNOMIAL PROJECTION



Key step: For $i = 1, \ldots, k$ we need to compute

$$\langle T_i, s \rangle = \int_{-1}^{1} s(x) T_i(x) dx$$

$$= \frac{1}{n} \sum_{j=1} T_i(\lambda_j) = \mathrm{tr}(T_i(\mathsf{A})).$$

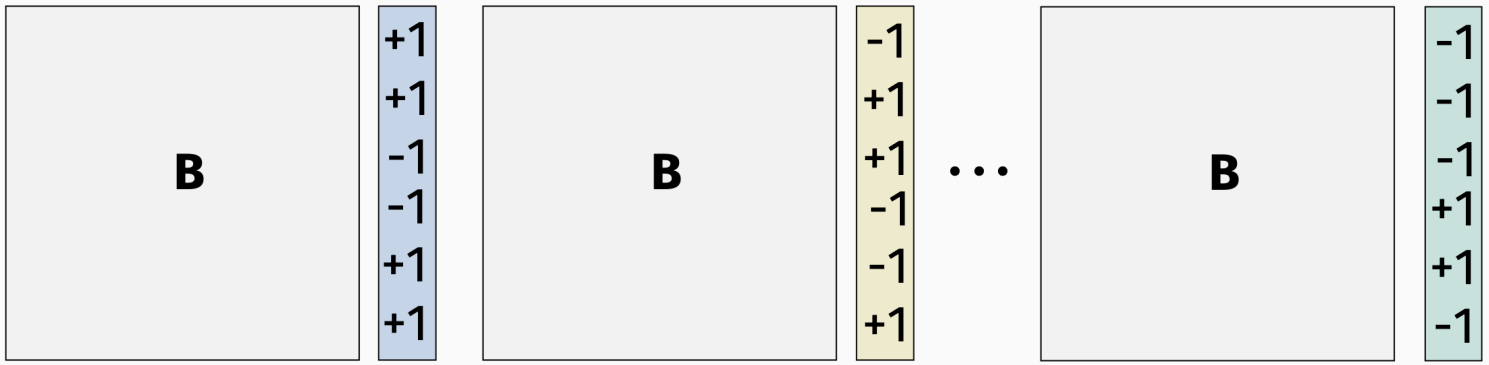
Goal: For $i = 1, \ldots, 1/\epsilon$, need to compute $\mathsf{tr}(T_i(\mathsf{A}))$.

Can be done very efficiently using a stochastic trace estimation algorithm!

We only require a small number of matrix-vector multiplications with $T_i(\mathsf{A})$.

Hutchinson 1991, Girard 1987:

- Draw $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ i.i.d. with random $\{+1, -1\}$ entries.
- Return $\tilde{T} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i^T \mathbf{B} \mathbf{x}_i$ as approximation to $\mathbf{tr}(\mathbf{B})$.



Requires m matvecs with $\mathbf{B}$.

Hutchinson's analysis

Let $\tilde{T}$ be the trace estimate returned by Hutchinson's method.

Claim (Rudelson, Vershynin, 2013, Roosta, Ascher 2015)

If $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, *then with probability* $(1-\delta)$,

$$\left|\tilde{T} - \mathrm{tr}(\mathbf{B})\right| \leq \epsilon \|\mathbf{B}\|_F.$$

Note that when $\mathbf{B}$'s eigenvalues lie between $[-1, 1]$, we have that $\|\mathbf{B}\|_F = O(\sqrt{n})$..

We can compute $\frac{1}{n}\,\mathbf{tr}(T_i(\mathbf{A}))$ up to additive error ϵ^2 using roughly:

$$\ell = \mathbf{min}\left(1, \frac{1}{n\epsilon^4}\right)$$

matrix-vector multiplies with $T_i(\mathbf{A})$.

Overall require $i\ell$ matrix-vector multiplies with A.

FULL KERNEL POLYNOMIAL METHOD

Theorem (Kernel Polynomial Method)

The Jackson-damped KPM provides an ϵ-approximate SDE with $O(\ell/\epsilon)$ matvecs with $\mathbf{A}$ *where* $\ell = \min\left(1, \frac{1}{n\epsilon^4}\right)$.

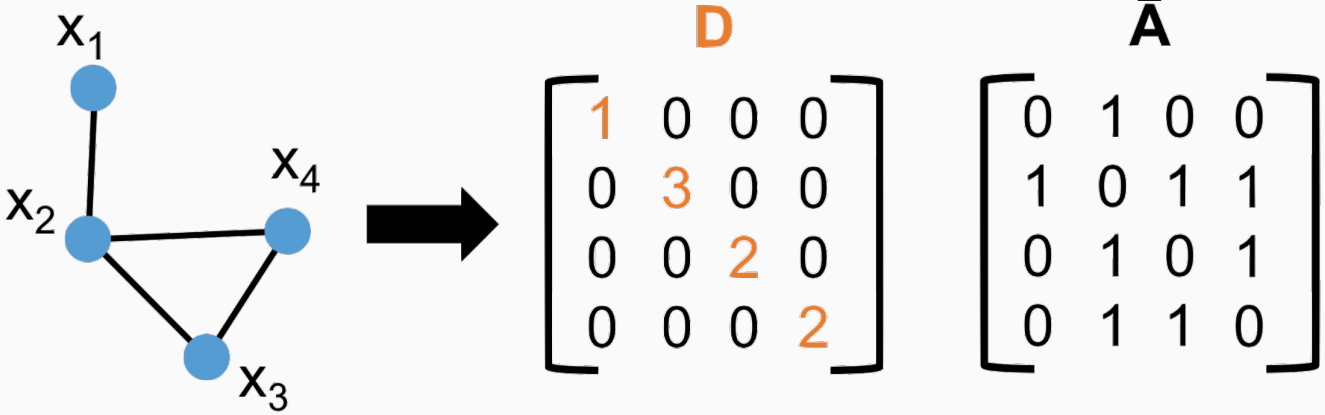
Theorem (Moment Matching Method)

A Chebyshev polynomial based MM-method provides an ϵ-approximate SDE with $O(\ell/\epsilon)$ matvecs with $\mathbf{A}$ *where* $\ell = \min\left(1, \frac{1}{n\epsilon^2}\right)$.

For typical values of n, ϵ, worst case running time is $O(n^2/\epsilon)$ for an $n \times n$ matrix A.

One recent method avoids the $O(n^2)$ cost for certain classes of matrices, running in sublinear time.

- [Cohen-Steiner, Kong, Sohler, Valiant, 2018] gives a method for normalized graph adjacency and Laplacian matrices assuming sample access to the graph.
- In $2^{O(1/\epsilon)}$ time returns ϵ-approximate spectrum. **Not practical, but very interesting!** No dependence on n.



Important in network science applications.

Uses a random walk based estimator to compute $\mathbf{tr}(\mathbf{A}^i)$ for $i = 1, \ldots, k$. Naturally interpretable as the chance of return after an i step random walk.

If we can compute $\mathbf{tr}(\mathbf{A}^i)$ for $i = 1, \ldots, k$, then we can of course compute $\mathbf{tr}(T_i(\mathbf{A}))$.

But this is a very poorly conditioned statement. Need to compute each $\mathbf{tr}(\mathbf{A}^i)$ to accuracy $\frac{1}{2}^{O(1/\epsilon)}$ to get an ϵ approximation to each $\mathbf{tr}(T_i(\mathbf{A}))$.

ALTERNATIVE APPROACH

Directly speed up computation of $\mathbf{tr}(T_i(\mathbf{A}))$. Recall that we required repeated matrix-vector multiplications with $T_i(\mathbf{A})$, which required matrix-vector multiplications with $\mathbf{A}$.

We can speed these up using random sampling!

A relatively coarse approximation is enough.

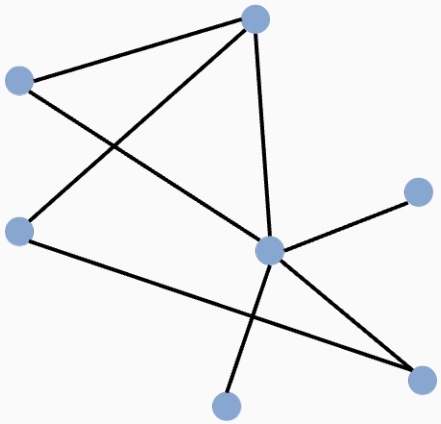
APPROXIMATE MATVECS FOR ADJACENCY MATRICES

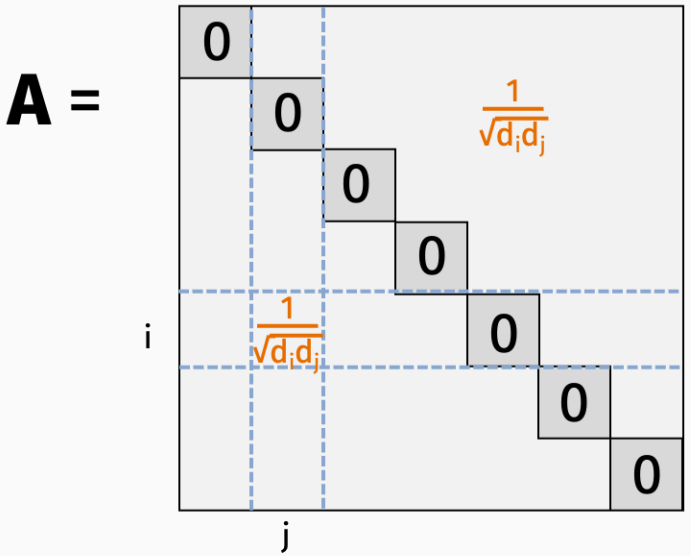
Claim (Approximate Matrix-Vector Multiplication)

There's an algorithm $\mathrm{AMV}(\mathbf{A}, \mathbf{x})$ *which, given sample access to any* $n \times n$ *normalized adjacency matrix* $\mathbf{A}$*, computes with high probability:*

$$\|\mathrm{AMV}(\mathbf{A}, \mathbf{x}) - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon \|\mathbf{x}\|_2.$$

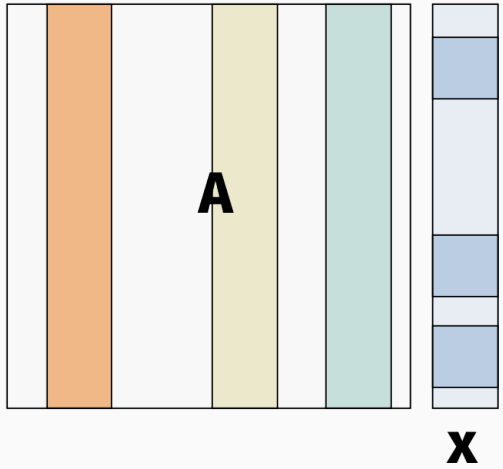
The algorithm runs in $O(n/\epsilon^2)$ *time.*





APPROXIMATE MATVECS FOR ADJACENCY MATRICES

Approximate $\mathbf{Ax}$ by randomly sampling columns proportional to ℓ_2 norm.



Drineas, Kannan, Mahoney, 2006. If we sample $O\left(\frac{1}{\Delta^2}\right)$ columns, each with probability proportional to

$$\frac{\|\mathbf{A}^i\|_2^2}{\|\mathbf{A}\|_F^2},$$

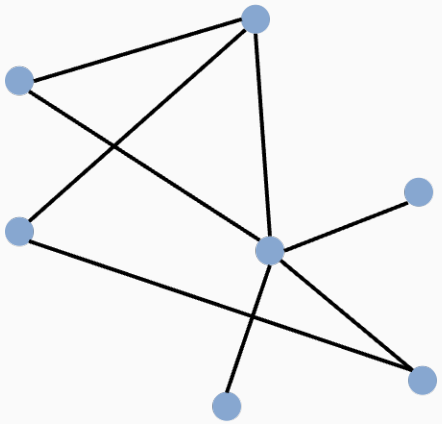
then with high probability:

$$\|\mathrm{AMV}(\mathbf{A}, \mathbf{x}) - \mathbf{Ax}\|_2 \leq \Delta \|\mathbf{A}\|_F \|\mathbf{x}\|_2.$$

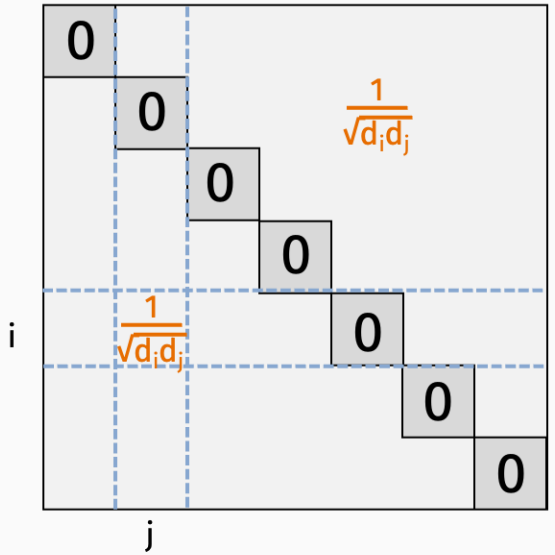
APPROXIMATE MATVECS FOR ADJACENCY MATRICES

Need to set $\Delta = \epsilon / \|\mathbf{A}\|_F$, which means that we will collect $\frac{\|\mathbf{A}\|_2^2}{\epsilon^2}$ samples.

Key Observation: Only columns corresponding to nodes with low-degree (i.e. sparse columns) get sampled with high probability. Dense columns are less likely to be sampled.



A =



IMPLEMENTING THE SAMPLING SCHEME

For node i with neighborhood $\mathcal{N}(i)$,

$$\|\mathbf{A}^i\|_2^2 = \sum_{j \in \mathcal{N}(i)} \frac{1}{d_i d_j}.$$

- Pick random node j.
- Pick random neighbor $i \in \mathcal{N}(j)$
- Sample column $\mathbf{A}^i$ with probability $1/d_i$.

Claim: With probability

$$\frac{1}{n} \sum_{j \in \mathcal{N}(i)} \frac{1}{d_i d_j} = \frac{1}{n} \|\mathbf{A}^i\|_2^2$$

we sample column $\mathbf{A}^i$. With some probability we get no sample.

Only get a sample with probability $\|\mathbf{A}\|_F^2/n$, so need to repeat the process $O(n/\epsilon^2)$ samples total.

What is the expected sparsity S of each column sampled?

$$\begin{aligned}\mathbb{E}[S] &= \sum_{i=1}^{n} d_i \frac{1}{n} \|\mathbf{A}^i\|_2^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} d_i \sum_{j \in \mathcal{N}(i)} \frac{1}{d_i d_j} \\ &= \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \frac{1}{d_j} = 1.\end{aligned}$$

So if we take $O(n/\epsilon^2)$ samples, it takes expected time $O(n/\epsilon^2)$ to compute an ϵ-approximate matrix-vector product!

PUTTING IT ALL TOGETHER

Our final goal is not to just multiply vectors by $\mathbf{A}$, but instead to multiply by $T_i(\mathbf{A})$.

One option is to use the three-term recurrence relation for Chebyshev polynomials:

- $\mathbf{v}_0 = \mathbf{x}$
- $\mathbf{v}_1 = \mathbf{A}\mathbf{x}$.
- For $k = 2, \ldots, i$,
    - $\mathbf{v}_k = 2\mathbf{A}\mathbf{v}_{k-1} - \mathbf{v}_{k-2}$.
- Return $\mathbf{v}_i$

How does error accumulate if we implement this recurrence with approximate matvecs?

- $\mathbf{v}_0 = \mathbf{x}$
- $\mathbf{v}_1 = \mathrm{AMV}(\mathbf{A}, \mathbf{x})$.
- For $k \geq 2$,
  - $\mathbf{v}_k = 2\mathrm{AMV}(\mathbf{A}, \mathbf{v}_{k-1}) - \mathbf{v}_{k-2}$.
- Return $\mathbf{v}_i$

Clenshaw showed, using an arguement based on Chebyshev polynomials of the second kind, that error builds quadratically. I.e. we can guarantee:

$$\|T_i(\mathbf{A})\mathbf{x} - \mathbf{v}_i\|_2 \leq (i^2) \cdot \epsilon \|\mathbf{x}\|_2$$

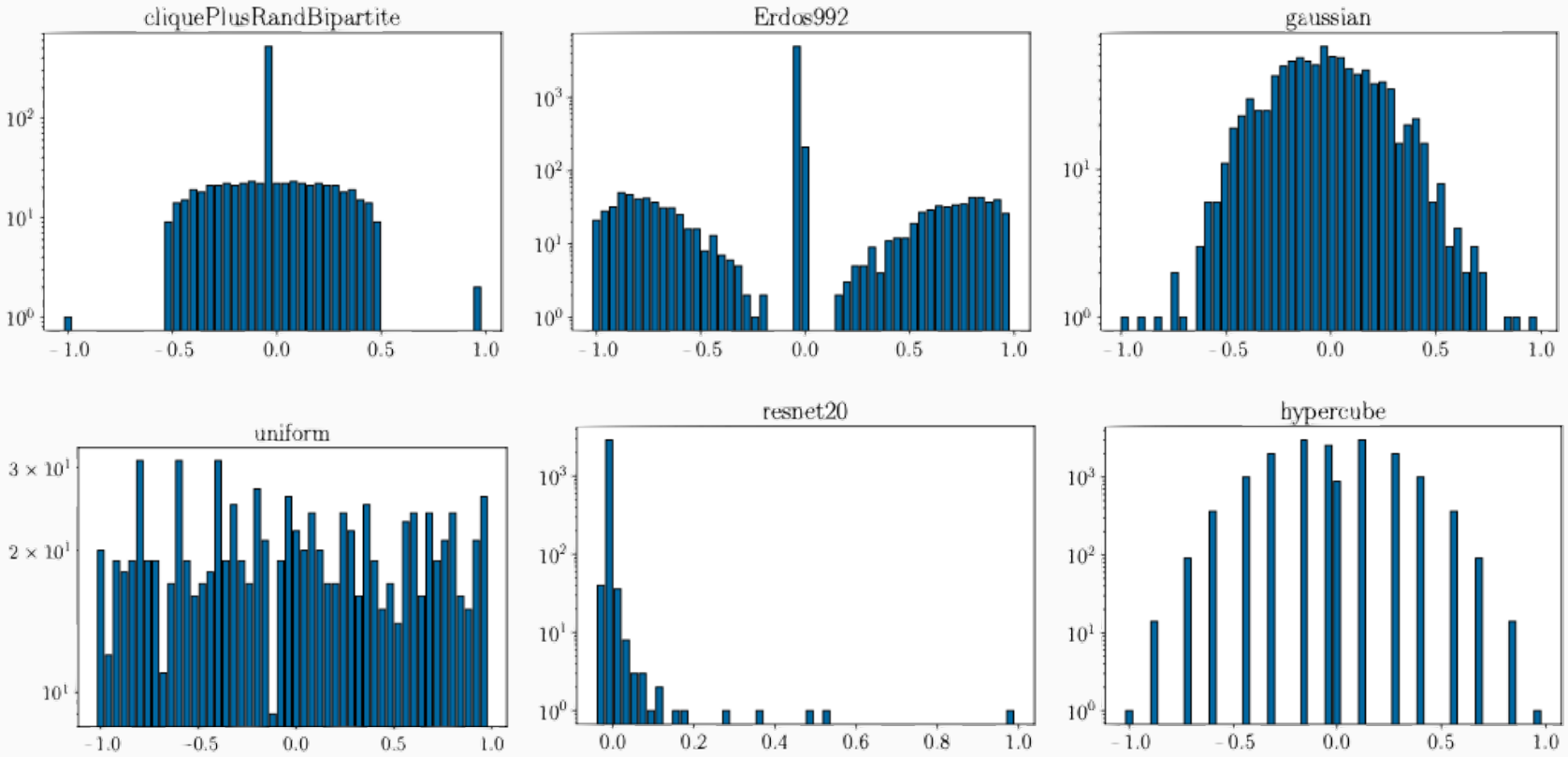
Theorem (Sublinear Time SDE)

It is possible to obtain and ϵ-approximate spectral density for a normalized graph adjacency matrix in $O(n/\epsilon^7)$ time.

OPEN QUESTIONS

- Improved ϵ dependence? Extension to weighted graphs?
- Compare to $2^{O(1/\epsilon)}$ time for Cohen-Steiner et al. method. Is $O(1/\epsilon^c)$ time possible for normalized adjacency matrices? $O(\sqrt{n}/\epsilon^c)$?
- Are there other classes of matrices where sublinear time results are possible?
- Is Wasserstein-1 distance the only metric we should care about?

How to handle structured spectra?



THANKS! QUESTIONS?

Alternative approach:

- Let $M_k(s) = [\langle s, T_0 \rangle, \langle s, T_1 \rangle, \ldots, \langle s, T_k \rangle]$ be a vector containing the first k Chebyshev moments of s.
- Find positive function q minimizing $\|M_k(s) - M_k(q)\|_1$.
- Can be solved efficiently using a small linear program or projected gradient descent.