Towards Optimal Matrix-Vector Complexity in Numerical Linear Algebra

By: Raphael A. Meyer

Advisor: Christopher Musco

Additional Committee Members:
Haim Avron, Chinmay Hegde, Jonathan Niles-Weed

Who am I? What is this thesis?

Theoretical Computer Scientist

Solve problems in Machine Learning, Data Science, Applied Math

Love working with Matrices

Research Areas:

Fast Numerical Linear Algebra

Todays' Defense

Out of scope

Randomized Functional Analysis

↳Robust Statistics

Responsible Data Science

↳Fairness & Explainability

Who am I? What is this thesis?

Theoretical Computer Scientist

Solve problems in Machine Learning, Data Science, Applied Math

Love working with Matrices

Research Areas:

Fast Numerical Linear Algebra

Todays' Defense

Out of scope

Randomized Functional Analysis

↳Robust Statistics

Responsible Data Science

↳Fairness & Explainability

Randomized Numerical Linear Algebra (RandNLA)

We can get this fast runtime if (ε, δ)-guarantee suffices:

① Return Approximate Solution — within 1% of true answer

② Use randomized algorithm — allowed to fail $\frac{1}{1000}$ times

At least $\frac{999}{1000}$ times you run ALGO, the result is within 1% of true value

This thesis: 4 papers in this literature

Outline

4 papers about 4 problems:

Trace Estimation	SOSA '20	10 min
Kronecker Trace Estimation	—	5 min
Low-Rank Approximation	SODA '24	25 min
$f(A)$ Low-Rank Approximation	—	5 min

Collaborators for these papers:

Chris Musco, Haim Avron, Cam Musco, David Persson, David Woodruff, Samson Zhou

Other collaborators:

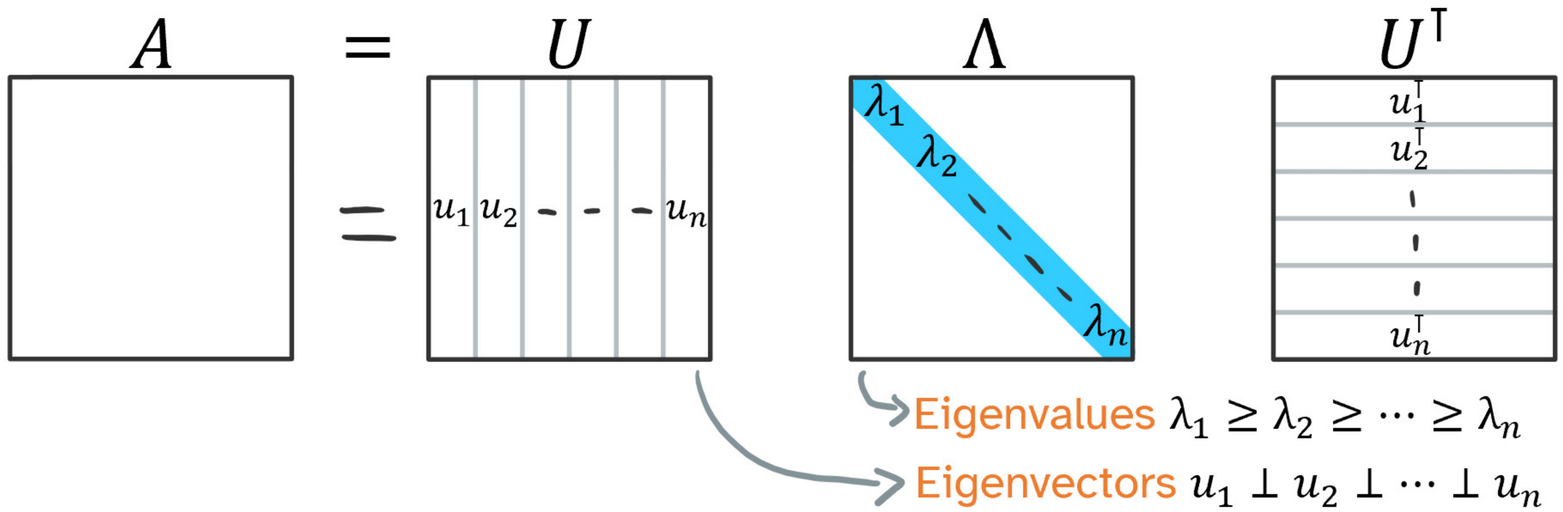
Jean Honorio, Amisha Jhanji, Falaah Arif Khan, Hemanta Maji, Venetia Pliatsika

Part 1:

Trace Estimation & Matrix-Vector Products

Linear Algebra Refresher

Let $A \in \mathbb{R}^{n \times n}$ be symmetric. Then $A = U \Lambda U^\intercal$



A is Positive Semi-Definite (PSD) if all $\lambda_i \geq 0$

Trace

$$tr(A) \coloneqq \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i$$

Frobenius Norm

$$\|A\|_F^2 \coloneqq \sum_{i,j=1}^{n} A_{ij}^2 = \sum_{i=1}^{n} \lambda_i^2 \overset{\text{For PSD } A}{\leq} (tr(A))^2$$

Motivation: Trace Estimation

Goal: Estimate the trace of matrix $A \in \mathbb{R}^{n \times n}$

$$tr(A) = \sum_{i=1}^{n} A_{ii} = \sum_{i}^{n} \lambda_i$$

In downstream applications, A is not stored in memory

Instead, B is in memory and $A = f(B)$:

No. Triangles $tr(\frac{1}{6}B^3)$	Estrada index $tr(e^B)$	Log-Determinant $tr(\ln(B))$

Computing B^3 takes $O(n^3)$ time SLOW

Computing $B^3x = B(B(Bx))$ takes $O(n^2)$ time FAST

If $A = f(B)$ then we can often compute Ax quickly

Can we estimate $tr(A)$ by computing $Ax_1, \ldots, Ax_m$?

Matrix-Vector Complexity

Formalize MatVecs as Computational Atom

Matrix-Vector Products

We can access $A \in \mathbb{R}^{n \times n}$ only via MatrixVectorOracle

i.e. a black box

$$x \xrightarrow{\text{input}} \text{ORACLE} \xrightarrow{\text{output}} Ax$$

"How many oracle queries to estimate $something(A)$?"

E.g. Random Projections, Power Method, Streaming, ...

[Simchowitz et al STOC '18, Sun et. al. ICALP '19, Braverman et. al. COLT '20, Bakshi et. al. STOC '22,]

Trace Estimation Problem

Estimate $tr(A)$ using as few MatVecs $Ax_1, \ldots, Ax_m$ as possible

$$\left|tr(A) - \widetilde{tr(A)}\right| \leq \varepsilon \, |tr(A)| \text{ w.p. } 1 - \delta$$

$\widetilde{tr(A)}$ within 1% error of $tr(A)$ at least $\frac{999}{1000}$ times

Girard-Hutchinson Trace Estimator

[Girard '87, Hutchinson '89, Avron Toledo '11, Roosta Ascher '15, Cortinovis Kressner 21]

- If $x \sim N(0, I)$, then

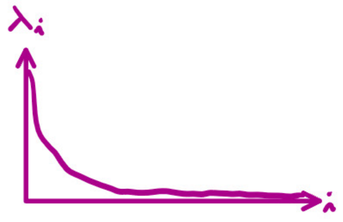
$$\mathbb{E}[x^\intercal A x] = tr(A) \qquad \mathrm{Var}[x^\intercal A x] = 2\|A\|_F^2$$

- Girard-Hutchinson: $H_\ell(A) := \frac{1}{\ell}\sum_{i=1}^{\ell} x_i^\intercal A x_i$

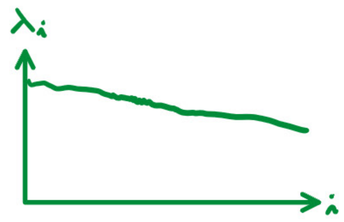
$$\mathbb{E}[H_\ell(A)] = tr(A) \qquad \mathrm{Var}[H_\ell(A)] = \frac{2}{\ell}\|A\|_F^2$$

Theorem: $H_\ell(A)$ needs $\ell = O(1/\varepsilon^2)$ for PSD A

But analysis only tight for nearly low-rank A



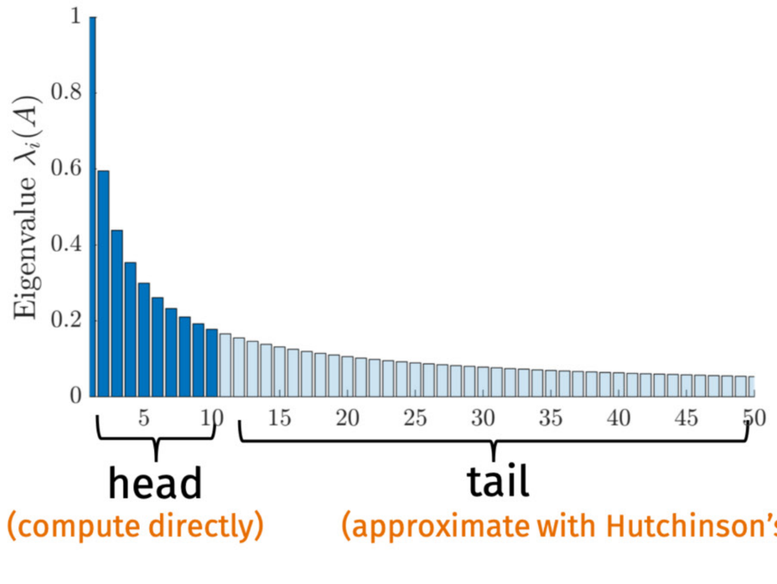
Nearly low-rank.
$\ell = O(1/\varepsilon^2)$ required.



Far from low-rank.
$\ell = O(1/\varepsilon)$ suffices.

Hutch++: Optimal Stochastic Trace Estimation

[**Meyer**, Musco, Musco, Woodruff SOSA '21]



5 Lines of Matlab!

Works great in practice!

For $\varepsilon = 0.01$,
10,000 Matvecs vs 100 Matvecs

Idea: Explicitly estimate top few eigenvalues. Use Girard-Hutchinson for the rest.

1. Find a good rank-k approximation $\widetilde{A_k}$ to A
2. Notice that $tr(A) = tr(\widetilde{A_k}) + tr(A - \widetilde{A_k})$
3. Compute $tr(\widetilde{A_k})$ exactly
4. Return $Hutch{+}{+}(A) \coloneqq tr(\tilde{A}_k) + H_\ell(A - \widetilde{A_k})$

Thm: taking $k = \ell = O(1/\varepsilon)$ suffices to converge in $O(1/\varepsilon)$ MatVecs!

People really like Hutch++!

Used in practice: implemented in SciPy!
↳Major Scientific Computing Library!

Publications on Better Implementations:

A-Hutch++ ↳[Persson, Cortinovis, Kressner '22]

Nystrom++ ↳[Persson, Cortinovis, Kressner '22]

Xtrace ↳[Epperly, Tropp, Webber '24]

Krylov-Aware ↳[Chen, Hallman '23]

Beyond MatVecs

Kronecker Trace Estimation ↳Next paper in this talk!

Aside: $\Omega(1/\varepsilon)$ lower bound in the Hutch++ paper!

Motivation: Modeling Quantum Physics

[Feldman et. al. '22]

Noa is a Quantum Physicist studying a grid of k (large) quantum particles, each particle "acts" in d dimensions

(d: a constant (like 2 or 8))

Matrix $A \in \mathbb{R}^{d^k \times d^k}$ describes how these particles act

Wants to compute Renyi Moments $tr(A^q)$ for integer q

Constraint: We can only efficiently compute *some* MatVecs

Can only efficiently compute Kronecker-MatVecs:

$$x_1, \ldots, x_k \in \mathbb{R}^d \xrightarrow{\text{input}} \text{ORACLE} \xrightarrow{\text{output}} A(x_1 \otimes \cdots \otimes x_k) \in \mathbb{R}^{d^k}$$

($\otimes$: Kronecker Product)

(or a compact representation thereof)

Hutchinson's Estimator is bad at Kronecker Trace Estimation

[**Meyer**, Avron '23 (preprint)]

How many Kronecker-MatVecs to estimate $tr(A)$?

If $x = x_1 \otimes \cdots \otimes x_k$ where $x_1, \ldots, x_k \sim N(0, I)$,

$$\mathbb{E}[x^\intercal A x] = tr(A)$$

$H_\ell(A)$ needs $\ell = \Theta(3^k/\varepsilon^2)$ in the worst case.

[Feldman et. al. '22, Avron et al '14, Bujanovic Kressner '21, Ahle et al '20]

Our Results:

Complex gaussian of variance 1

If $x_j = \frac{1}{\sqrt{2}}(y_j + iz_j)$ for $y_j, z_j \sim N(0, I)$, then $\ell = \Theta\left(2^k/\varepsilon^2\right)$ instead

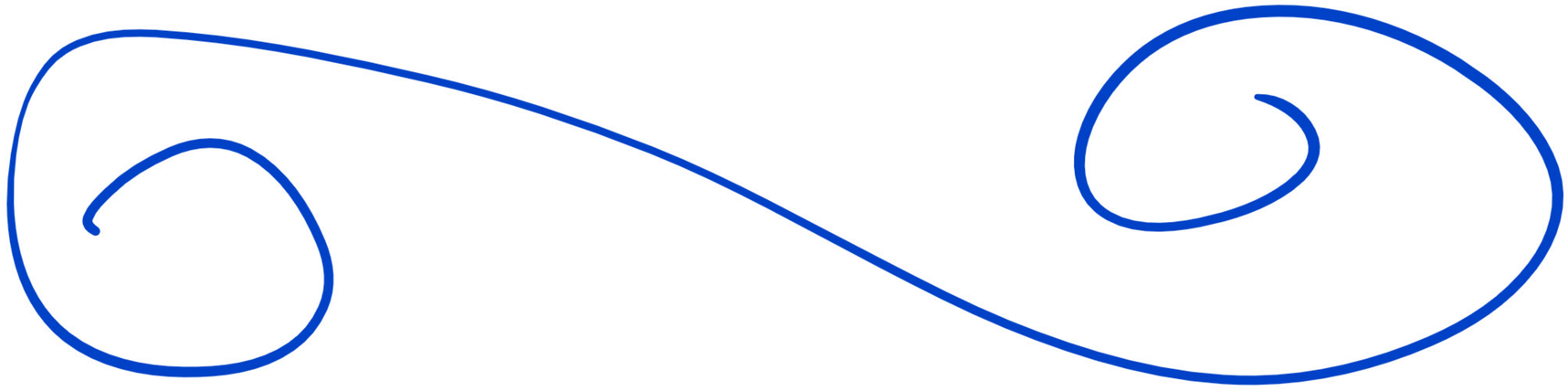
Exact variance of $H_\ell(A)$ *it's ugly though*

Key Technique: nothing fancy *(first principles + induction)*

First ever Kronecker-MatVec lower bound: $\Omega(\sqrt{k}/\varepsilon)$

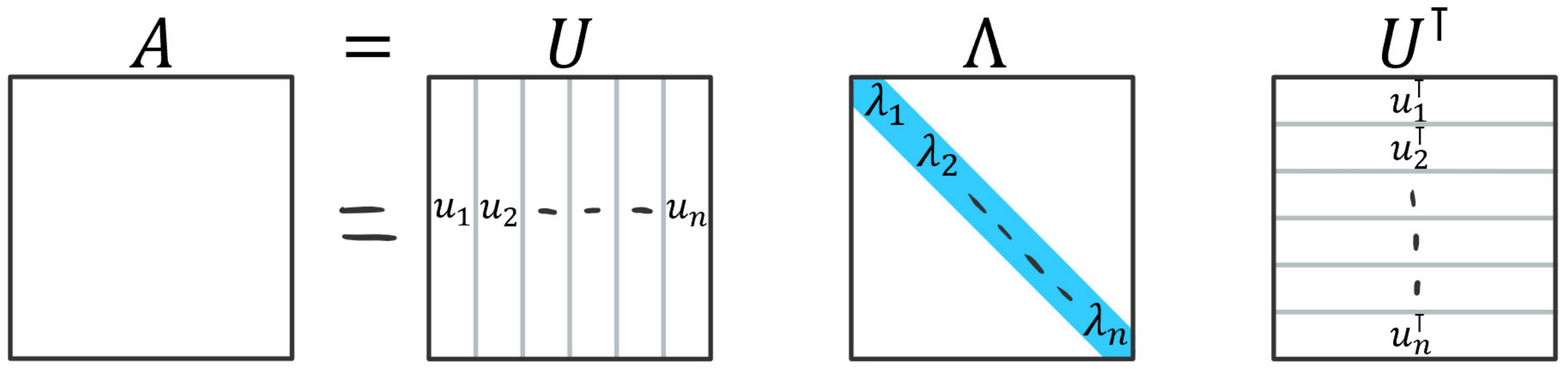
Part 2:

Low-Rank Approximation



Linear Algebra Refresher pt. 2

Let $A \in \mathbb{R}^{n \times n}$ be symmetric. Then $A = U \Lambda U^\intercal$



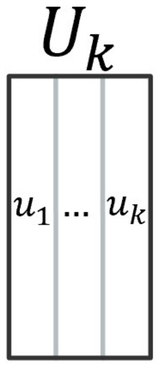
Eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

Eigenvectors $u_1 \perp u_2 \perp \cdots \perp u_n$

A is Positive Semi-Definite (PSD) if all $\lambda_i \geq 0$

For PSD A, $\underset{rank(B) \leq k}{\operatorname{argmin}} \|A - B\| = U_k U_k^\intercal A$

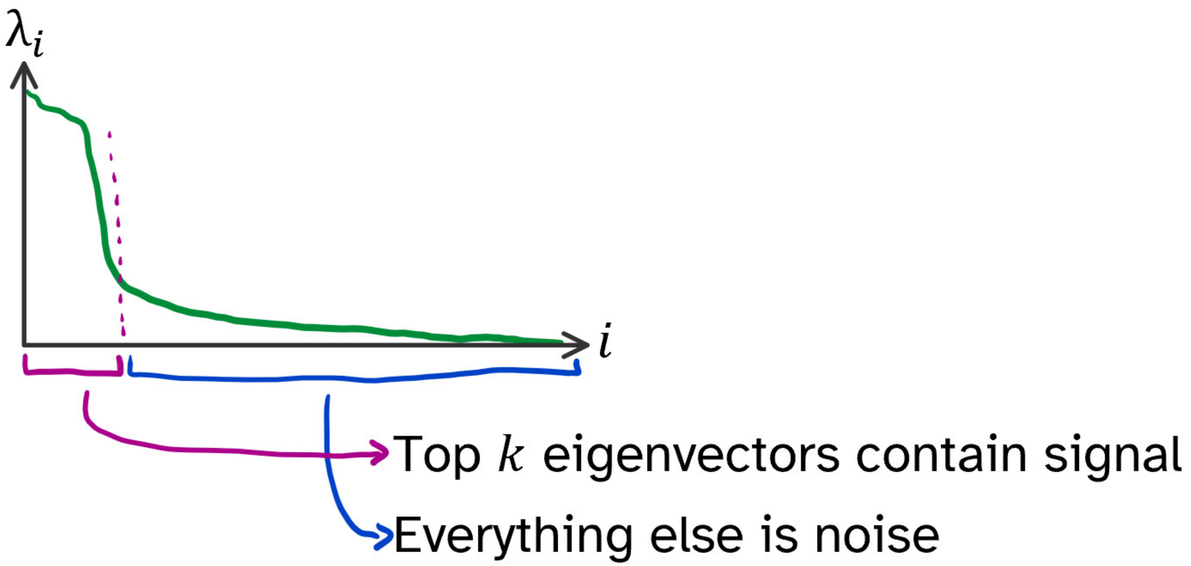
[Eckart-Young-Mirsky Thm]



Motivation: Matrix Denoising

Alice has a very large matrix $A \in \mathbb{R}^{n \times n}$

Most of that data is noisy:



Goal: keep only the k good directions in the data

Find the best rank-k approximation to A

People *really* care about this problem

[Drineas Mahoney Cristianini '05, Rokhlin et. al. '09, Mahoney Drineas '09, Halko Martinsson Tropp '11, Drineas Ipsen '14, Gu '15, Clarkson Woodruff '17, Tropp '18, Yuan '18, Bakshi Clarkson Woodruff '22, ...]

Low-Rank Approximation

Given $A \in \mathbb{R}^{n \times n}$ (PSD), $k, \varepsilon > 0$ find $Q \in \mathbb{R}^{n \times k}$ with ortho cols s.t.

$$\left\| A - QQ^{\intercal}A \right\| \leq (1 + \varepsilon) \min_{rank(B) \leq k} \|A - B\|$$

Ideally, $Q = U_k$, so use a Block Krylov to approx U_k

[Rokhlin et al. '09, Halko et al. '11, Drineas Ipsen '19, Tropp Webber '23, ...]

Block Krylov

1. Pick a start block

   $B \in \mathbb{R}^{n \times b}$ Usually Gaussian

   $b =$ block size

2. Build Krylov subspace

   $Z = \text{orth}([B \;\; AB \;\; A^2B \;\; \ldots \;\; A^tB])$ ↝ t = iteration complexity

3. Return the best approx to A in $\text{span}(Z)$

   $Q = Z^\intercal V_k$ where V_k = top k eigvecs of $Z^\intercal A A^\intercal Z$

Uses $O(bt)$ MatVecs. How should we pick b?

Contribution: first theorems showing that $b = 1$ is a good idea

How should we pick b?

1. Large block size $b \geq k$

Rich line of work [Tropp, Halko, Martinson, Gu, Drineas, Ipsen, Woodruff, ...]

Strong theoretical results for L.R.A. specifically

Gap-Independent Convergence [Musco Musco '15]

$$b = k, B_{ij} \sim N(0,1) \quad \Rightarrow \quad t = O\left(\frac{1}{\sqrt{\varepsilon}} \log\left(\frac{n}{\varepsilon}\right)\right)$$

$\tilde{O}(k/\sqrt{\varepsilon})$ matvecs

Exponential Convergence

$t = \tilde{O}\left(\log\left(\frac{n}{\varepsilon}\right)\right)$ suffices for A fast eigeval decay

$\tilde{O}(k \log(1/\varepsilon))$ matvecs

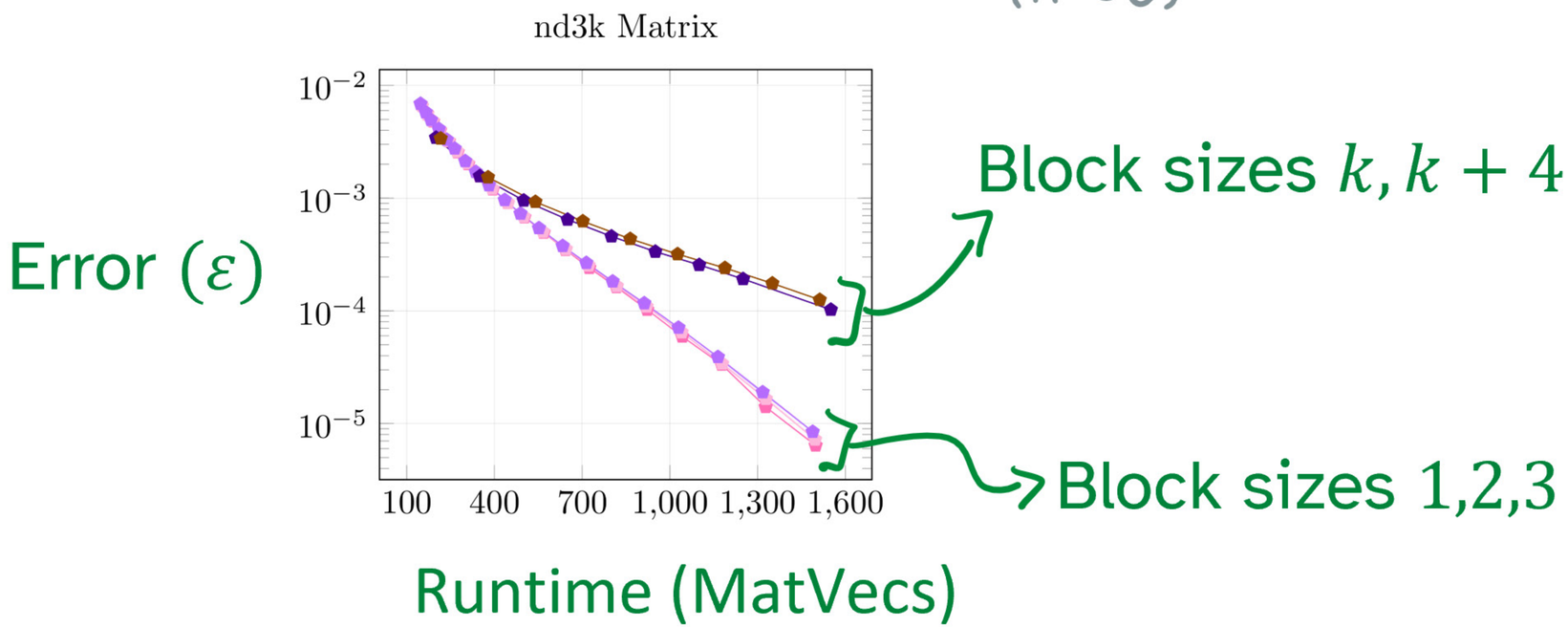
How should we pick b?

2. Small block size $\boldsymbol{b \ll k}$

$b = 1$ is called "Single Vector Krylov"
Applied Math suggests $b \approx$ size of eigval clusters
Lack results* for Low-Rank Approximation
Cannot be gap-independent

In practice, $b = 1$ often just works well *(MATLAB does this!)*

$(K=50)$



A theory/practice gap!

When and why do small block methods match/outperform large block methods for low-rank approximation?

Caveat: Infinite Precision, Matrix-vector Complexity

Main Result

For all $\ell \geq k$,

Number of MatVecs needed by Single Vector Krylov

is less than

Number of MatVecs needed by block size ℓ Krylov

If any $\ell \geq k$ is fast, then single vector is fast

Up to log dependence on eigengaps

Proof by Black-box-ish Reduction

Main Result (Rigorous)

Let g_{min} = smallest gap between any of top k eigs

$$= \min_{i \in \{1,\ldots,k-1\}} \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1}}$$

Then,

$b = 1$ converges with $t = O\left(\frac{k}{\sqrt{\varepsilon}} \log\left(\frac{1}{g_{min}}\right) + \frac{1}{\sqrt{\varepsilon}} \log\left(\frac{n}{\varepsilon}\right)\right)$

$\tilde{O}\left(k/\sqrt{\varepsilon}\right)$ matvecs

Main Result 2 (Semi-Rigorous)

If A has fast eigval decay,

[Musco, Musco 15] $\tilde{O}\left(k \log\left(\frac{1}{\varepsilon}\right)\right)$

[**Meyer**, Musco, Musco 24] $\tilde{O}\left(k \log\left(\frac{1}{g_{min}}\right) + \log\left(\frac{1}{\varepsilon}\right)\right)$

Upside: separate k from ε

Downside: depends on g_{min}

Key Observation: A Silly Manipulation

Suppose $b = 1$, so for $x \sim N(0, I)$

$$Z = \text{orth}([\underline{x}\ A\underline{x}\ A^2\underline{x} \cdots A^t\underline{x}])$$

Now, repeat some columns

$$= \text{orth}([\underline{x}\ A\underline{x} \cdots A^{\ell}\underline{x}\ \ A\underline{x}\ A^2\underline{x} \cdots A^{\ell+1}\underline{x}\ \ A^2\underline{x}\ A^3\underline{x} \cdots A^{\ell+2}\underline{x}\ \ \cdots\ \ A^{t-\ell}\underline{x} \cdots A^t\underline{x}])$$

$$= \text{orth}([\ S_\ell \qquad AS_\ell \qquad A^2S_\ell \quad \cdots \quad A^{t-\ell}S_\ell\ ])$$

Where $S_\ell = [\underline{x}\ A\underline{x} \cdots A^{\ell}\underline{x}]$ is our Simulated Start Block

$b = 1$ Krylov Subspace
for t iterations
starting from $x \sim \mathrm{N}(0, \mathrm{I})$

$\approx$

$b = \ell$ Krylov Subspace
for $t - \ell$ iterations
starting from S_ℓ

Upside: 1 matvec = 1 iteration of block Krylov

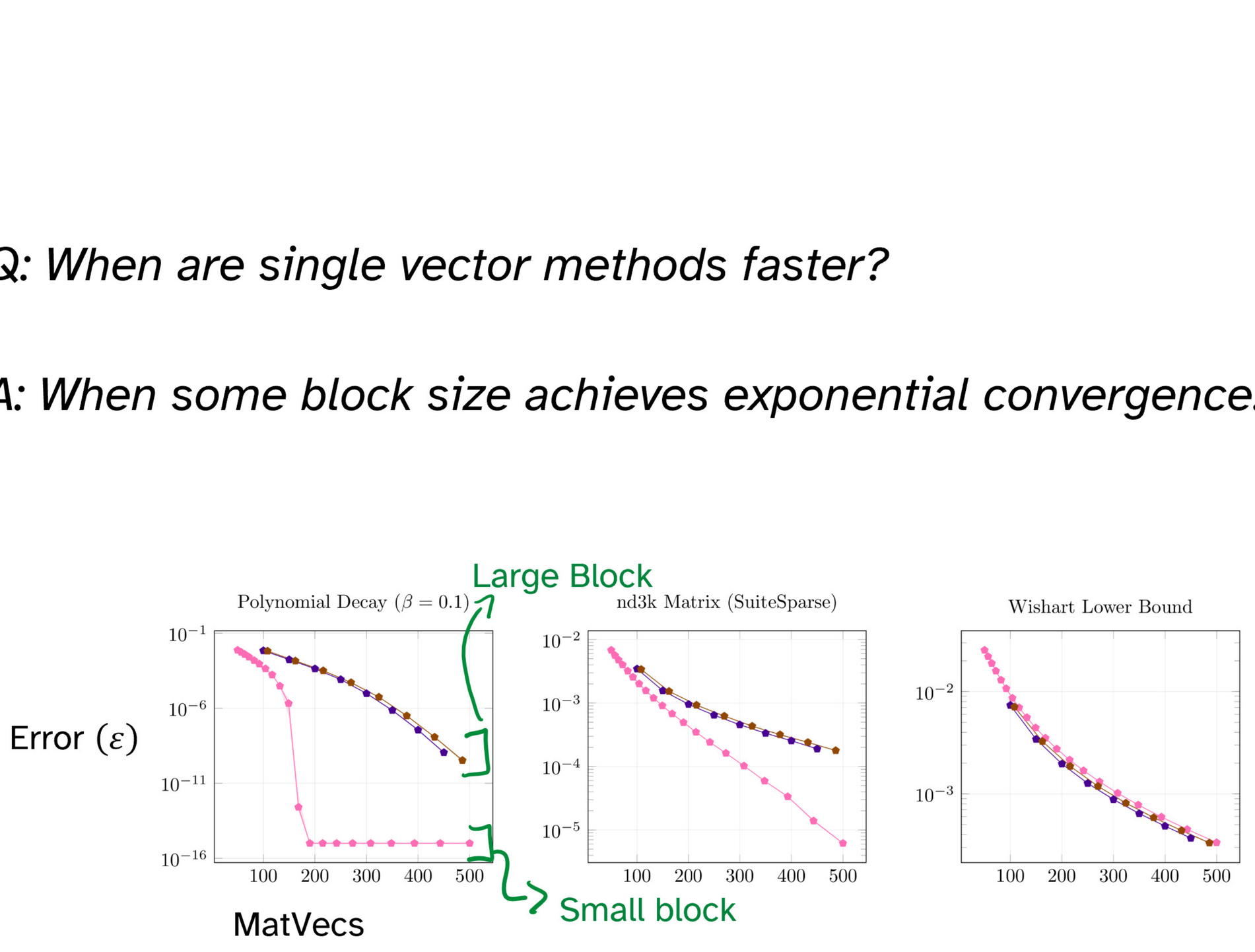
Downside: S_ℓ is a bad starting block $\quad S_\ell = [x\ Ax \ldots A^\ell x]$

Thm [Musco, Musco '15] if S_ℓ is "L-good", then $t = O\left(\frac{1}{\sqrt{\varepsilon}} \log \frac{nL}{\varepsilon}\right)$ suffices

Thm [**Meyer**, Musco, Musco '24] S_ℓ is L-good for $L = \left(g_{min}^k \cdot poly(n/\varepsilon)\right)$

Q: When are single vector methods faster?

A: When some block size achieves exponential convergence.



Conclusion *(for Single Vector Krylov)*

In this talk:

Proof intuition for block size 1

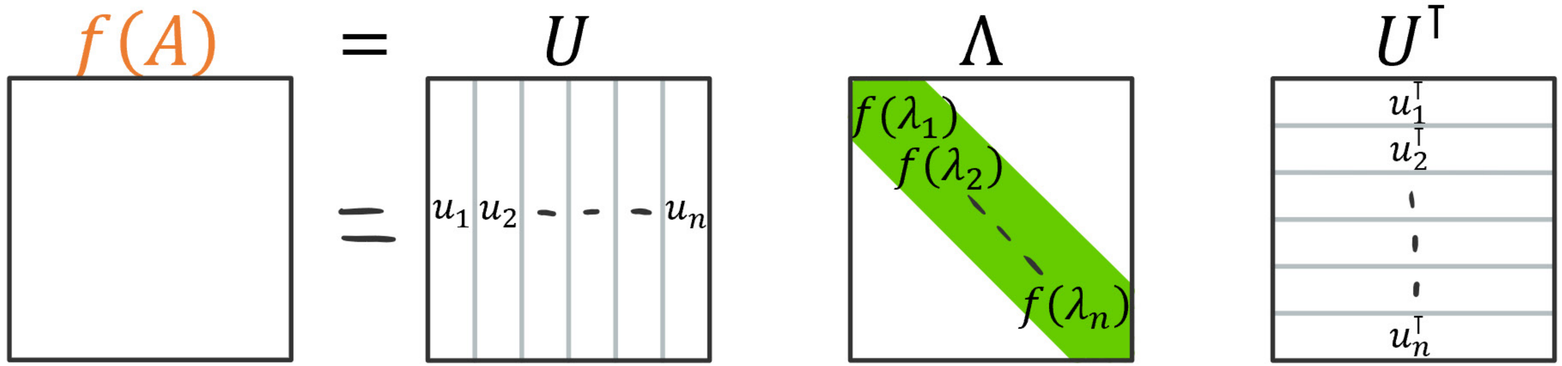
In the paper: Grab bag of more implications

- Beyond $b = 1$
- **Smoothed Analysis shatters** g_{min}
- Simplify Fast-Frobenius L.R.A. [Bakshi et al. '22]
- Faster-ish Schatten-norm L.R.A
- Single Vector Subspace Iteration
- Experiments

Final Topic: $f(A)$ Low-Rank Approximation

Let $A \in \mathbb{R}^{n \times n}$ PSD, $f: \mathbb{R} \to \mathbb{R}$ monotonically increasing, $f(0) = 0$

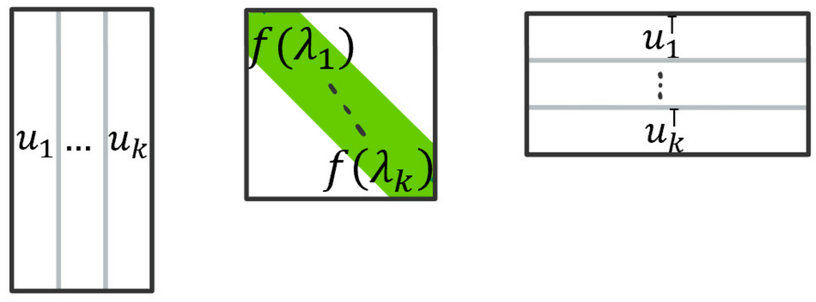
Then $A = U\Lambda U^\intercal$ has



↳ Eigenvalues $f(\lambda_1) \geq f(\lambda_2) \geq \cdots \geq f(\lambda_n)$

Sometimes we want a low-rank approximation to $f(A)$!

$$\big(f(A)\big)_k = U_k \quad f(\Lambda_k) \quad U_k^\intercal = f(A_k)$$



funNystrom Algorithm [Persson Kressner '23]

Idea: LowRankApprox to $f(A) = f(\text{LowRankApprox to } A)$ for increasing $f, f(0) = 0$

① Find $Q \in \mathbb{R}^{n \times k}$ with ortho cols, $\|A - QQ^\intercal A\| \leq (1+\varepsilon)\|A - A_k\|$

② Build $B = AQ(Q^\intercal AQ)^+ Q^\intercal A$ Nystrom Approx, is PSD

③ Return $C = f(B)$ Uses that B is PSD, so $f(B)$ well defined

Super fast! Basically same MatVec complexity as finding Q!

Algorithm-Agnostic Low-Rank Approximation of Operator Monotone Matrix Functions

[Persson **Meyer** Musco '23]

Prior work [Persson Kressner '23]

If $Q \in \mathbb{R}^{n \times k}$ is found using SubspaceIter, then $C \approx \big(f(A)\big)_k$

Only for "Operator Monotone" f (read: smooth concave)

Proves convergence from scratch

Our Contribution: *funNystrom is Algorithm-Agnostic*

Suppose we have $Q \in \mathbb{R}^{n \times k}$ s.t. $\|A - QQ^\intercal A\| \leq (1+\varepsilon)\|A - A_k\|$

Then Nystrom Approx also has $\|A - B\| \leq (1+\varepsilon)\|A - A_k\|$

Then funNystrom Approx also has $\|f(A) - C\| \leq (1+\varepsilon)\|f(A) - f(A_k)\|$

↳ *f operator monotone*

Algorithm Agnostic! Same ε! Simple proofs!

Part 3:

Conclusion

Conclusion

Progress on 4 fundamental NLA Tasks:

Trace Estimation	[**Meyer** Musco Musco Woodruff SOSA '21]
Kronecker Trace Estimation	[**Meyer** Avron '23 (preprint)]
Low-Rank Approximation	[**Meyer** Musco Musco SODA '24]
$f(A)$ Low Rank Approximation	[Persson **Meyer** Musco '23 (preprint)]

Broad Themes:

Matrix-Vector Complexity

Simple proof techniques

Pragmatic Algorithms

Thesis of the Thesis

1. Matrix-Vector Complexity
2. Simple proof techniques
3. Pragmatic Algorithms

Progress on Fundamental
NLA Problems is still possible,
without sophisticated new techniques!

THANK You!