Randomized Algorithms for Solving Linear Systems with Low-Rank Structure

Michał Dereziński
Computer Science and Engineering, University of Michigan

Based on joint works with Zachary Frangella, Daniel LeJeune, Christopher Musco, Deanna Needell, Pratik Rathore, Elizaveta Rebrova, Aaron Sidford, Madeleine Udell, and Jiaming Yang

ISL Colloquium, Stanford University

October 2, 2025

Outline

Example. Solve this system of **linear equations**:

$$\begin{aligned} 3x + 2y + z &= 39 \\ 2x + 3y + z &= 34 \\ x + 2y + 3z &= 26 \end{aligned}$$

Example. Solve this system of **linear equations**:

$$\begin{aligned} 3x + 2y + z &= 39 \\ 2x + 3y + z &= 34 \\ x + 2y + 3z &= 26 \end{aligned}$$

Solution: **Method of elimination** (a.k.a. Gaussian elimination)

First appeared in:
The Nine Chapters on the Mathematical Art,
China, 2nd century

Later formalized by:
Newton, and then Gauss, among others.

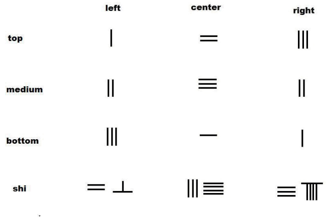
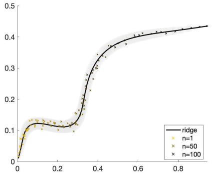


Image by Gisling, CC BY 3.0, url.

Solving linear systems in modern applications

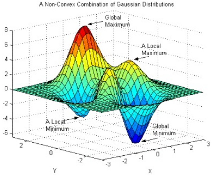
Statistical inference

$$y_i = f^*(\mathbf{x}_i) + \xi_i, \qquad f^* \in \mathcal{F}, \quad (\mathbf{x}_i, y_i) \sim \mathcal{D}.$$



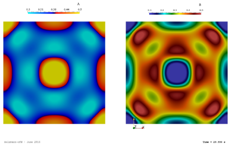
Nonlinear optimization

$$\text{Minimize} \quad f(\mathbf{x}) = \sum_{i=1}^{n} \log(1 + \mathrm{e}^{-y_i \mathbf{a}_i^\top \mathbf{x}_i})$$



Partial differential equations

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u)$$



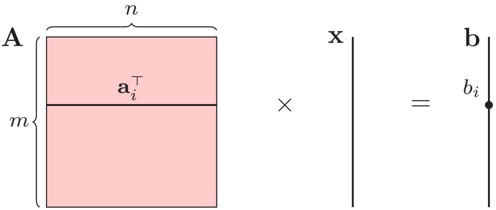
What is the cost of solving an $m \times n$ linear system?

Answer: It depends...

- ... on the sparsity of the input matrix,
- ... on its singular value decay profile,
- ... on whether it has domain-specific structural properties (e.g., Laplacian, Hankel, Toeplitz, circulant matrices, etc.),
- ... on whether we need exact, or high/machine precision, or medium/low precision,
- ... on whether we can tolerate a small chance of failure.

Two perspectives on solving linear systems

Task: Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^n$, solve a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$.
For simplicity, assume that the system is consistent and $m = O(n)$.



Direct methods

- Method of elimination
- QR, LU, SVD, ...

$O(n^3)$ time

for "ill-conditioned systems"

Iterative methods

- Conjugate gradient (CG)
- MINRES, GMRES, LSQR, ...

$O(n^2)$ time × T iterations

for "well-conditioned systems"

Can we unify the two perspectives?

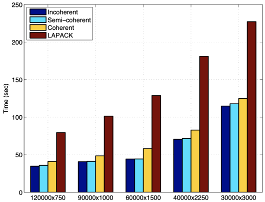
Key ingredient: Randomization

Randomization

Using **randomized algorithms** to solve **deterministic problems**.

Early successes in NLA [AMT10]

*“**beats LAPACK’s** direct dense **least-squares** solver …on essentially any dense **tall matrix.**”*



Moving towards “RandLAPACK”

Murray et al. *“Randomized Numerical Linear Algebra: A Perspective on the Field with an Eye to Software,”* arXiv:2302.11474, 2023.

[AMT10] Avron, Maymounkov, and Toledo. “Blendenpik: Supercharging LAPACK’s least-squares solver”, *SIAM Journal on Scientific Computing* 32.3: 1217-1236, 2010.

This talk: Randomization for general linear systems

- Low-rank structure in ill-conditioned systems

  In "typical" systems, a low-rank component causes ill-conditioning.

- Randomized algorithms for low-rank structure

  Randomized sketching-based methods can go beyond traditional (Krylov-based) methods for solving low-rank structured systems.

- Case study: Solving a dense $10^8 \times 10^8$ linear system

  Scaling full Kernel Ridge Regression to massive datasets.

- Unified perspective on the complexity of solving linear systems:

$$\tilde{O}\Big(\underbrace{k^3}_{\text{low-rank}} + \underbrace{n^2}_{\text{well-conditioned}} \Big)$$

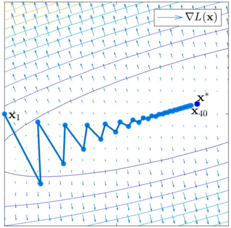
Outline

Convergence of iterative methods

Task: Given $\mathbf{A} \in \mathbb{R}^{O(n)\times n}$ and $\mathbf{b} \in \mathbb{R}^n$, solve a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$

Iterative methods

- Conjugate gradient (CG)
- MINRES, GMRES, LSQR, ...



$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2) \text{ operations}} \quad \times \quad T \text{ iterations}$$

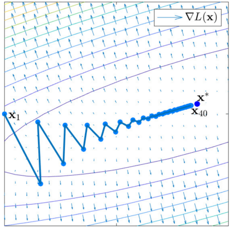
How does the number of iterations T depend on $\mathbf{A}$?

Convergence of iterative methods

Task: Given $\mathbf{A} \in \mathbb{R}^{O(n)\times n}$ and $\mathbf{b} \in \mathbb{R}^n$, solve a linear system $\mathbf{Ax} = \mathbf{b}$

Iterative methods

- Conjugate gradient (CG)
- MINRES, GMRES, LSQR, ...



$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{Av}}_{O(n^2) \text{ operations}} \quad \times \quad T \text{ iterations}$

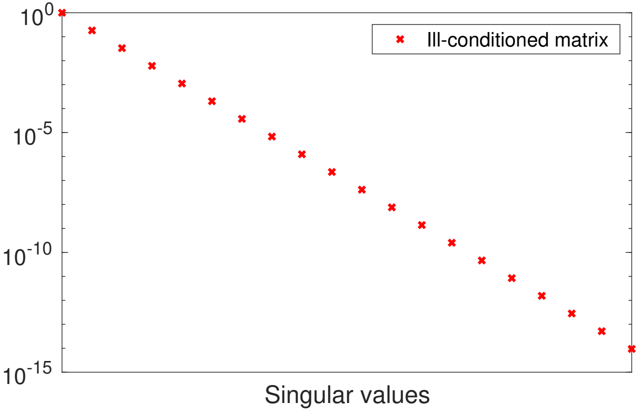
How does the number of iterations T depend on $\mathbf{A}$?

Naïve answer: T scales with the *condition number*, $\kappa = \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}$.

$\Rightarrow$ *"Iterative methods perform poorly for ill-conditioned systems."*

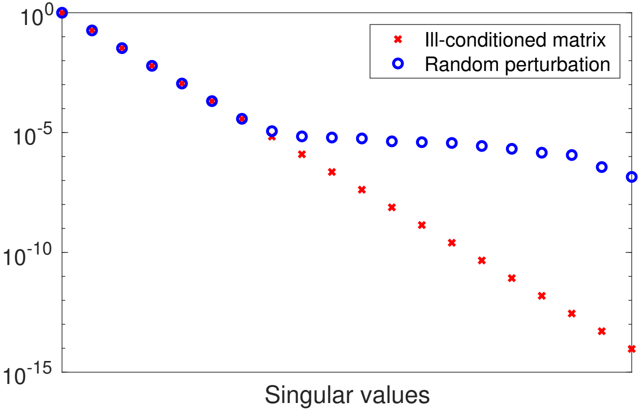
Typical ill-conditioned system

What does the spectrum of a "typical" ill-conditioned matrix look like?



Typical ill-conditioned system

What does the spectrum of a "typical" ill-conditioned matrix look like?

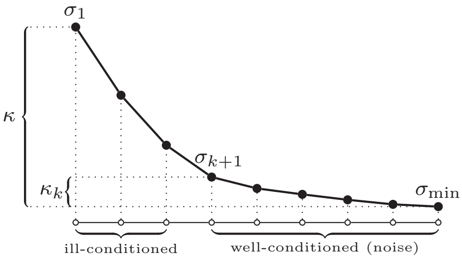


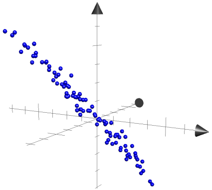
Random perturbation: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{G}, \qquad \|\mathbf{G}\| \leq 10^{-5}\|\mathbf{A}\|$

Model: Systems with low-rank structure

Low-rank structure: Implicitly partition the spectrum of $\mathbf{A}$

1. Ill-conditioned top-k: favors *direct* methods
2. Well-conditioned tail: favors *iterative* methods





Systems with low-rank structure are ubiquitous across many areas!

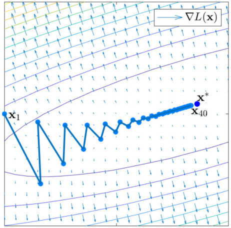
- "Signal + noise" data, e.g., smoothed analysis, stochastic rounding, ...
- Deliberate regularization in ML/Stats/Opt, e.g., $\mathbf{A} = \mathbf{B} + \lambda\mathbf{I}$
- Key subroutine in matrix norm and eigenvalue estimation methods

Back to convergence of iterative methods

Task: Given $\mathbf{A} \in \mathbb{R}^{O(n)\times n}$ and $\mathbf{b} \in \mathbb{R}^n$, solve a linear system $\mathbf{Ax} = \mathbf{b}$

Iterative methods

- Conjugate gradient (CG)
- MINRES, GMRES, LSQR, ...



$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{Av}}_{O(n^2)\text{ operations}} \quad \times \quad T \text{ iterations}$$

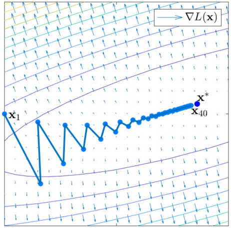
How do they perform on ***systems with low-rank structure****?*

Back to convergence of iterative methods

Task: Given $\mathbf{A} \in \mathbb{R}^{O(n)\times n}$ and $\mathbf{b} \in \mathbb{R}^n$, solve a linear system $\mathbf{Ax} = \mathbf{b}$

Iterative methods

- **Conjugate gradient (CG)**
- **MINRES, GMRES, LSQR, ...**



$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{Av}}_{O(n^2) \text{ operations}} \quad \times \quad T \text{ iterations}$

How do they perform on ***systems with low-rank structure****?*

Answer: Convergence theory of **Krylov Subspace Methods**

Krylov subspaces and polynomial approximation

Definition: Given square matrix $\mathbf{A}$ and vector $\mathbf{b}$, the order-k Krylov subspace is defined as:

$$\mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \text{span}\left\{\mathbf{b},\ \mathbf{A}\mathbf{b},\ \mathbf{A}^2\mathbf{b},\ \ldots,\ \mathbf{A}^{k-1}\mathbf{b}\right\}.$$

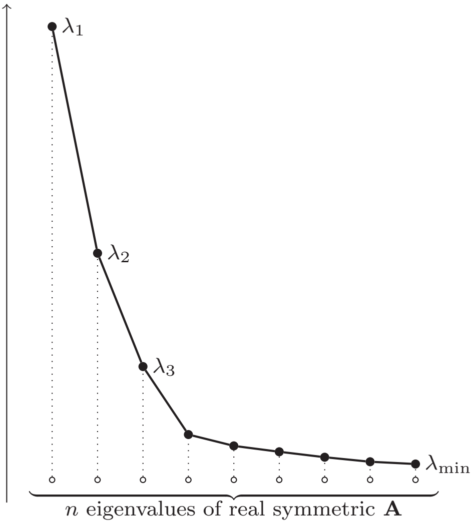
Property: Any vector $\mathbf{v} \in \mathcal{K}_k(\mathbf{A}, \mathbf{b})$ can be expressed as $\mathbf{v} = p(\mathbf{A})\mathbf{b}$, where $p(x) = c_0 + c_1 x \ldots + c_{k-1} x^{k-1}$ is a polynomial of degree $k-1$.

Recipe for linear solver: Gradually build a Krylov subspace and maintain "best" approximation $\hat{\mathbf{x}} = p(\mathbf{A})\mathbf{b}$ for $\mathbf{A}^{-1}\mathbf{b}$ in that subspace.

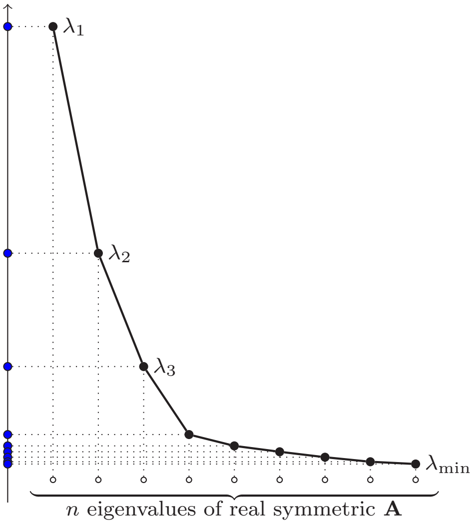
Dominant cost: Matrix-vector product to compute the next direction,

$$\mathbf{A}^k\mathbf{b} = \mathbf{A} \cdot (\mathbf{A}^{k-1}\mathbf{b}) \quad \text{in } O(n^2) \text{ arithmetic operations.}$$

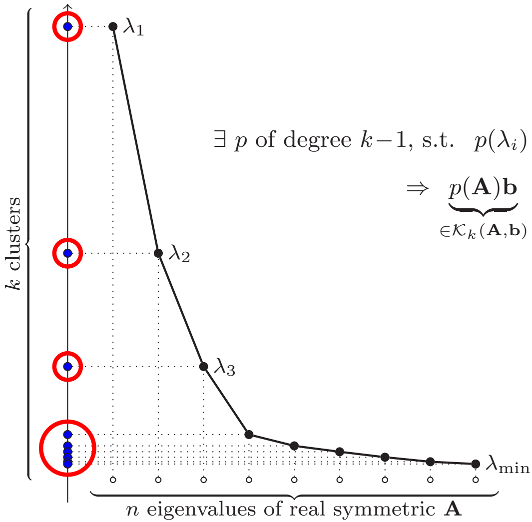
Krylov subspaces and eigenvalue clusters



Krylov subspaces and eigenvalue clusters



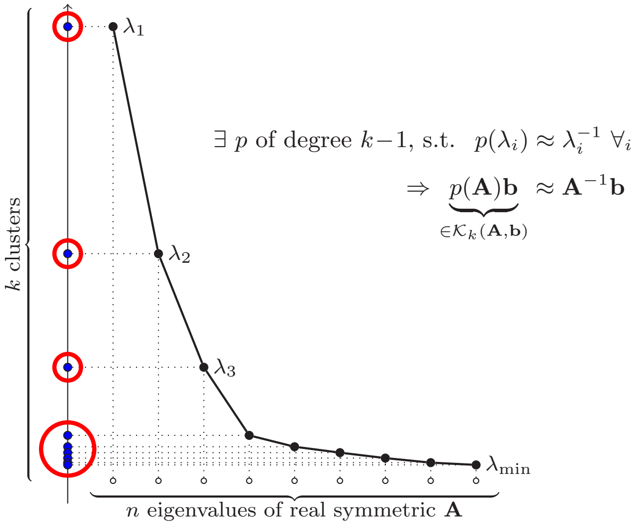
Krylov subspaces and eigenvalue clusters



$$\exists\ p \text{ of degree } k-1, \text{ s.t. } \quad p(\lambda_i) \approx \lambda_i^{-1}\ \forall_i$$

$$\Rightarrow \underbrace{p(\mathbf{A})\mathbf{b}}_{\in \mathcal{K}_k(\mathbf{A},\mathbf{b})} \approx \mathbf{A}^{-1}\mathbf{b}$$

Krylov subspaces and eigenvalue clusters



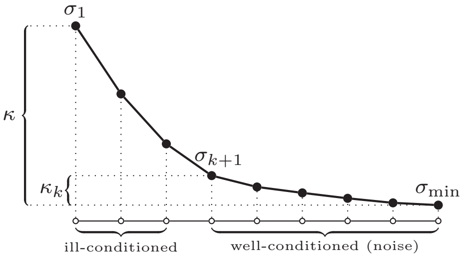
Conclusion: If the eigenvalues of $\mathbf{A}$ form k clusters, then $\mathcal{K}_k(\mathbf{A}, \mathbf{b})$ contains an accurate approximation to $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$.

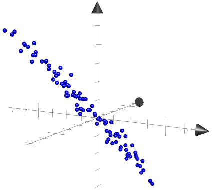
Krylov subspace methods: Convergence

Theorem ([AL86])

If $\mathbf{A}$ *has singular values* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min}$ *with* $\kappa_k = \frac{\sigma_{k+1}}{\sigma_{\min}}$, *there is a Krylov subspace method (e.g., LSQR) that finds an* ϵ*-approximate solution* $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\| \leq \epsilon\|\mathbf{b}\|$ *in*

$$T = O(k + \kappa_k \log 1/\epsilon) \quad \textit{iterations.}$$





(We use singular values instead of eigenvalues to capture the non-symmetric case.)

[AL86] Axelsson and Lindskog. "On the rate of convergence of the preconditioned conjugate gradient method," *Numerische Mathematik*. 48:499-523, 1986.

Krylov subspace methods and low-rank structure

Conclusion: To solve an $O(n) \times n$ linear system with at most k large isolated singular values and a cluster of width κ_k, we need:

$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2) \text{ operations}} \times O(k + \kappa_k \log 1/\epsilon) \;=\; O(n^2 k + n^2 \kappa_k \log 1/\epsilon)$$

Krylov subspace methods and low-rank structure

Conclusion: To solve an $O(n) \times n$ linear system with at most k large isolated singular values and a cluster of width κ_k, we need:

$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2) \text{ operations}} \times O(k + \kappa_k \log 1/\epsilon) \;=\; O({\color{red}n^2 k} + n^2 \kappa_k \log 1/\epsilon)$$

Question: Can we avoid the $n^2 k$ bottleneck in Krylov methods?

Krylov subspace methods and low-rank structure

Conclusion: To solve an $O(n) \times n$ linear system with at most k large isolated singular values and a cluster of width κ_k, we need:

$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2)\text{ operations}} \times O(k + \kappa_k \log 1/\epsilon) \;=\; O(n^2 k + n^2 \kappa_k \log 1/\epsilon)$$

Question: Can we avoid the $n^2 k$ bottleneck in Krylov methods?

Yes! and **No!**

Krylov subspace methods and low-rank structure

Conclusion: To solve an $O(n) \times n$ linear system with at most k large isolated singular values and a cluster of width κ_k, we need:

$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2) \text{ operations}} \times O(k + \kappa_k \log 1/\epsilon) \;=\; O(n^2 k + n^2 \kappa_k \log 1/\epsilon)$$

Question: Can we avoid the $n^2 k$ bottleneck in Krylov methods?

Yes! and **No!**

With only $\mathbf{v} \to \mathbf{A}\mathbf{v}$ access to $\mathbf{A}$,

Krylov methods are optimal.

Krylov subspace methods and low-rank structure

Conclusion: To solve an $O(n) \times n$ linear system with at most k large isolated singular values and a cluster of width κ_k, we need:

$$\underbrace{\text{Cost of } \mathbf{v} \to \mathbf{A}\mathbf{v}}_{O(n^2) \text{ operations}} \times O(k + \kappa_k \log 1/\epsilon) \;=\; O(n^2 k + n^2 \kappa_k \log 1/\epsilon)$$

Question: Can we avoid the $n^2 k$ bottleneck in Krylov methods?

Yes! and **No!**

With *direct* access to $\mathbf{A}$, randomized methods do better.

With only $\mathbf{v} \to \mathbf{A}\mathbf{v}$ access to $\mathbf{A}$, Krylov methods are optimal.

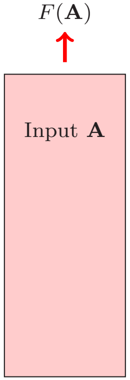
Our result: $\tilde{O}(k^3 + n^2 \kappa_k \log 1/\epsilon)$

Outline

Outline

Main tool: Randomized sketching

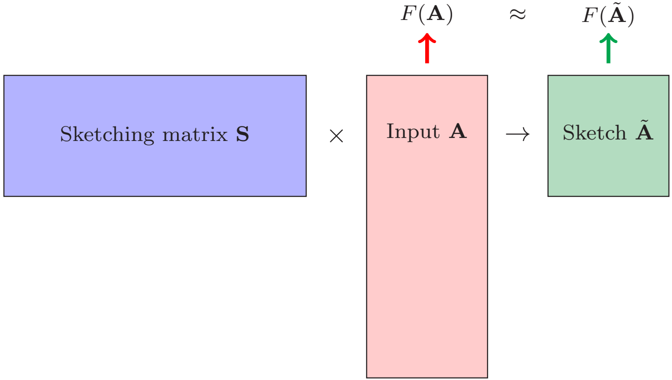
Sketching operator: Random dimension reducing linear map (matrix $\mathbf{S}$)
E.g.: Gaussian/sparse matrices, randomized Hadamard transforms, ...



Crucially: Can be *much* faster than dense matrix multiplication

Main tool: Randomized sketching

Sketching operator: Random dimension reducing linear map (matrix $\mathbf{S}$)
E.g.: Gaussian/sparse matrices, randomized Hadamard transforms, ...

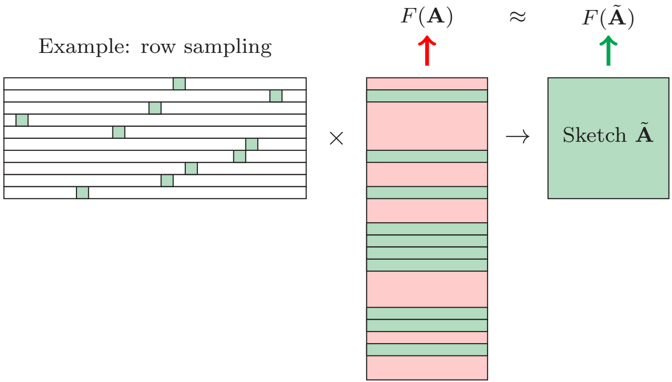


Crucially: Can be *much* faster than dense matrix multiplication

Main tool: Randomized sketching

Sketching operator: Random dimension reducing linear map (matrix $\mathbf{S}$)

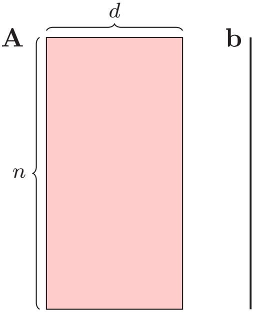
E.g.: Gaussian/sparse matrices, randomized Hadamard transforms, ...



Crucially: Can be *much* faster than dense matrix multiplication

Example: Least squares

Over-determined linear system: Many more equations than unknowns, also known as *least squares*.



$$\text{Compute} \quad \mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

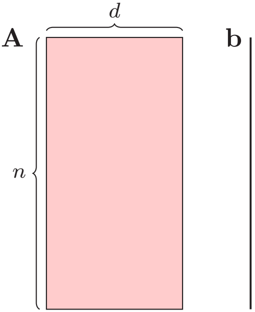
$$\text{for} \quad \mathbf{A} \in \mathbb{R}^{n \times d}, \ \mathbf{b} \in \mathbb{R}^n$$

Example: Least squares

Over-determined linear system: Many more equations than unknowns, also known as *least squares*.

Compute $\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$

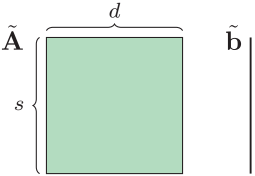
for $\mathbf{A} \in \mathbb{R}^{n \times d}, \ \mathbf{b} \in \mathbb{R}^n$



Sketching leverages this redundancy:

Compute $\widetilde{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\tilde{\mathbf{A}}\mathbf{x} - \tilde{\mathbf{b}}\|_2^2$

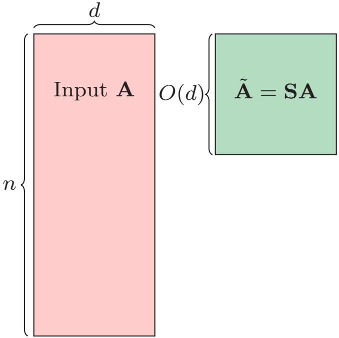
for $\underbrace{\tilde{\mathbf{A}} = \mathbf{S}\mathbf{A}}_{\text{sketch of } \mathbf{A}}, \ \underbrace{\tilde{\mathbf{b}} = \mathbf{S}\mathbf{b}}_{\text{sketch of } \mathbf{b}}$



Why does sketching work for least squares?

Fact. In $\tilde{O}(nd)$ time, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(d) \times d}$ such that for $i = 1, ..., d$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A})$$



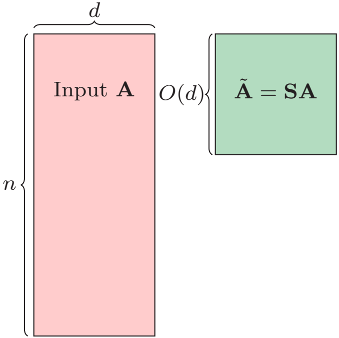
Why does sketching work for least squares?

Fact. In $\tilde{O}(nd)$ time, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(d)\times d}$ such that for $i = 1, ..., d$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A})$$

Even better. This preserves the whole covariance structure of the matrix:

$$\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}} = (1 \pm \tfrac{1}{2})\mathbf{A}^\top \mathbf{A}.$$



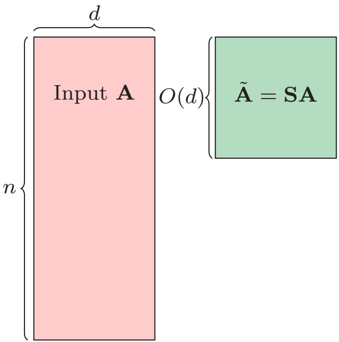
Why does sketching work for least squares?

Fact. In $\tilde{O}(nd)$ time, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(d)\times d}$ such that for $i = 1, ..., d$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A})$$

Even better. This preserves the whole covariance structure of the matrix:

$$\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}} = (1 \pm \tfrac{1}{2})\mathbf{A}^\top \mathbf{A}.$$

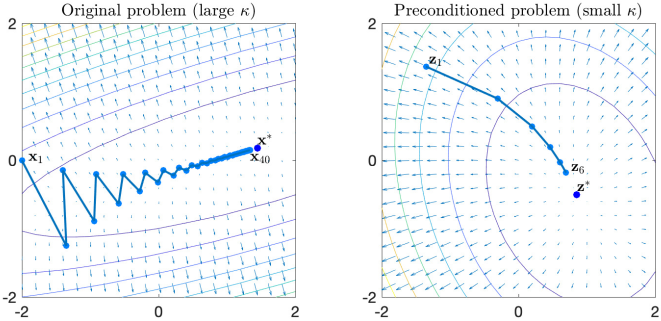


Fast least squares solver: Runs in $\tilde{O}(d^3 + nd)$ time

1. Rewrite least squares via normal equations, $\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$
2. Precondition your favorite iterative method with $\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}}$

Fast least squares solver

<u>Compute preconditioner:</u> $(\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^{-1} \approx (\mathbf{A}^\top \mathbf{A})^{-1}$ Cost: $O(d^3)$



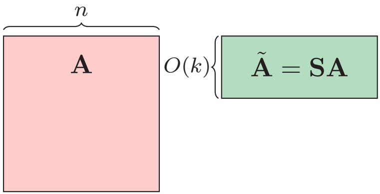
$$\mathbf{A}^\top \mathbf{A} \mathbf{x} = \mathbf{A}^\top \mathbf{b}$$

$\kappa(\mathbf{A}^\top \mathbf{A})$ is large

$$\mathbf{A}^\top \mathbf{A} (\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^{-1} \mathbf{z} = \mathbf{b}$$

$\kappa(\mathbf{A}^\top \mathbf{A} (\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^{-1})$ is small

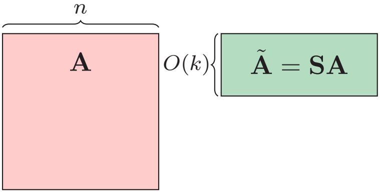
Does this work for general linear systems?



Does this work for general linear systems?

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, ..., k$:

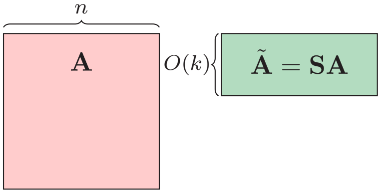
$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



Does this work for general linear systems?

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, \dots, k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$

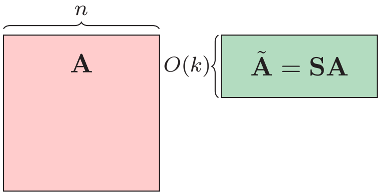


Bottom line: The sketch picks up **noise** from the **spectral tail**
$\Rightarrow$ weak preconditioner (both in theory and in practice)

Does this work for general linear systems?

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k) \times n}$ such that for $i = 1, ..., k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k} \sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



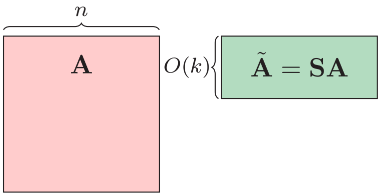
Bottom line: The sketch picks up **noise** from the **spectral tail**
$\Rightarrow$ weak preconditioner (both in theory and in practice)

Can we still produce a fast solver via sketching?

Does this work for general linear systems?

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, ..., k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



Bottom line: The sketch picks up **noise** from the **spectral tail**
$\Rightarrow$ weak preconditioner (both in theory and in practice)

Can we still produce a fast solver via sketching?

Yes! Just use multiple sketches:

1. Sketch-and-Project
2. Recursive Sketching

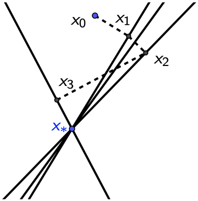
Outline

Background: The Kaczmarz algorithm

Idea: Iteratively project onto the solutions of individual equations.

Starting at $\mathbf{x}_0$, for $t = 0, 1, 2, \ldots$

1. Select index I_t
2. Project current iterate $\mathbf{x}_t$ onto the solutions of I_t-th equation



Randomized Kaczmarz: Select indices via weighted sampling [SV09]

The first Kaczmarz algorithm with provable convergence rate.

[Kac37] Stefan Kaczmarz, "Angenäherte Auflösung von Systemen linearer Gleichungen", *Bulletin International de l'Académie Polonaise des Sciences et des Lettres* 35:355–357, 1937.

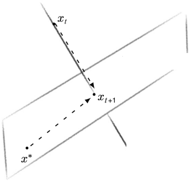
[SV09] Strohmer and Vershynin, "A randomized Kaczmarz algorithm with exponential convergence", *Journal of Fourier Analysis and Applications*, 14.2:262-278, 2009.

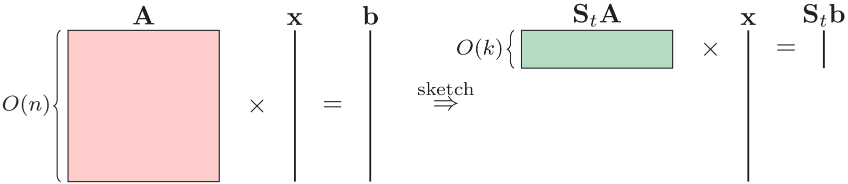
Powerful extension: Sketch-and-Project

Starting at $\mathbf{x}_0 \in \mathbb{R}^n$, for $t = 0, 1, 2, ...$

1. Sample random $O(k) \times O(n)$ matrix $\mathbf{S}_t$.
2. Project $\mathbf{x}_t$ onto the solutions of $\mathbf{S}_t\mathbf{A}\mathbf{x} = \mathbf{S}_t\mathbf{b}$:

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{x}_t - \mathbf{x}\|^2 \quad \text{subject to} \quad \mathbf{S}_t\mathbf{A}\mathbf{x} = \mathbf{S}_t\mathbf{b}.$$





[GR15] Gower and Richtárik, "Randomized iterative methods for linear systems", *SIAM Journal on Matrix Analysis and Applications*, 36.4:1660-1690, 2015.

Key insight: Sharp analysis of Sketch-and-Project

Theorem ([DR24])

If $\mathbf{S}_t$ is a Gaussian matrix, then Sketch-and-Project satisfies:

$$\mathbb{E}\,\|\mathbf{x}_t - \mathbf{x}^*\|^2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\frac{1}{k}\sum_{i>k}\sigma_i^2(\mathbf{A})}\right)^t \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

[DR24] Dereziński and Rebrova, "Sharp Analysis of Sketch-and-Project Methods via a Connection to Randomized Singular Value Decomposition", *SIAM Journal on Mathematics of Data Science*, 6.1:127-153, 2024.

Key insight: Sharp analysis of Sketch-and-Project

Theorem ([DR24])

If $\mathbf{S}_t$ *is a Gaussian matrix, then Sketch-and-Project satisfies:*

$$\mathbb{E}\,\|\mathbf{x}_t - \mathbf{x}^*\|^2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\frac{1}{k}\sum_{i>k}\sigma_i^2(\mathbf{A})}\right)^t \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

We again observe the ***tail noise*** *from sketching.*

[DR24] Dereziński and Rebrova, "Sharp Analysis of Sketch-and-Project Methods via a Connection to Randomized Singular Value Decomposition", *SIAM Journal on Mathematics of Data Science*, 6.1:127-153, 2024.

Key insight: Sharp analysis of Sketch-and-Project

Theorem ([DR24])

If $\mathbf{S}_t$ is a Gaussian matrix, then Sketch-and-Project satisfies:

$$\mathbb{E}\,\|\mathbf{x}_t - \mathbf{x}^*\|^2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\frac{1}{k}\sum_{i>k}\sigma_i^2(\mathbf{A})}\right)^t \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

We again observe the ***tail noise*** *from sketching.*

Key insight:

- Each sketch-and-project step runs on a $\frac{k}{n}$**-fraction** of the data
- This runtime gain should cancel out the **tail noise** $\leq \frac{n}{k}\sigma_{k+1}^2$

[DR24] Dereziński and Rebrova, "Sharp Analysis of Sketch-and-Project Methods via a Connection to Randomized Singular Value Decomposition", *SIAM Journal on Mathematics of Data Science*, 6.1:127-153, 2024.

Key insight: Sharp analysis of Sketch-and-Project

Theorem ([DR24])

If $\mathbf{S}_t$ is a Gaussian matrix, then Sketch-and-Project satisfies:

$$\mathbb{E}\,\|\mathbf{x}_t - \mathbf{x}^*\|^2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\frac{1}{k}\sum_{i>k}\sigma_i^2(\mathbf{A})}\right)^t \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

We again observe the ***tail noise*** *from sketching.*

Key insight:

- Each sketch-and-project step runs on a $\cancel{k}/\cancel{n}$**-fraction** of the data
- This runtime gain should cancel out the **tail noise** $\leq \cancel{n}/\cancel{k}\,\sigma_{k+1}^2$

[DR24] Dereziński and Rebrova, "Sharp Analysis of Sketch-and-Project Methods via a Connection to Randomized Singular Value Decomposition", *SIAM Journal on Mathematics of Data Science*, 6.1:127-153, 2024.

Key insight: Sharp analysis of Sketch-and-Project

Theorem ([DR24])

If $\mathbf{S}_t$ is a Gaussian matrix, then Sketch-and-Project satisfies:

$$\mathbb{E}\,\|\mathbf{x}_t - \mathbf{x}^*\|^2 \;\leq\; \left(1 - \frac{\sigma_{\min}^2(\mathbf{A})}{\frac{1}{k}\sum_{i>k}\sigma_i^2(\mathbf{A})}\right)^t \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

We again observe the ***tail noise*** *from sketching.*

Key insight:

- Each sketch-and-project step runs on a $\cancel{\frac{k}{n}}$-**fraction** of the data
- This runtime gain should cancel out the **tail noise** $\leq \cancel{\frac{n}{k}}\sigma_{k+1}^2$

Hang on, Gaussian sketching is still too expensive!

[DR24] Dereziński and Rebrova, "Sharp Analysis of Sketch-and-Project Methods via a Connection to Randomized Singular Value Decomposition", *SIAM Journal on Mathematics of Data Science*, 6.1:127-153, 2024.

Making this efficient

Advances in sketch-and-project for systems with low-rank structure:

1. Sketching: *Gaussian guarantees for fast sketches*
   - Randomized Hadamard transform [DY24]
   - Leverage score sampling [RFY$^+$25]

2. Projecting: *Fast computation of the projection step*
   - Fast inner solver using PCG [DY24]
   - Amortizing projection cost across iterations [DNRY25]

3. Acceleration: *Improved convergence using momentum*
   - Convergence analysis with Nesterov's momentum [DLNR24]
   - Adaptive tuning of momentum parameters [DNRY25]

Making this efficient: Sketch-and-Project++

Theorem ([DNRY25])

Given any k, we can solve an $O(n) \times n$ linear system $\mathbf{Ax} = \mathbf{b}$ in time:

$$\tilde{O}\big(nk^2 + n^2\kappa_k \log 1/\epsilon\big), \quad \textit{where} \quad \kappa_k = \frac{\sigma_{k+1}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}.$$

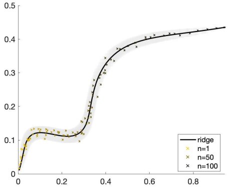
- Overcomes the n^2k bottleneck of Krylov subspace methods.
- Sketch-and-Project++: New family of randomized linear solvers
  Implementations: Kaczmarz++/CD++ [DNRY25], ASkotch [RFY$^+$25]

(Still does not attain the promised $k^3 + n^2$ guarantee...)

[DNRY25] Dereziński, Needell, Rebrova, and Yang. *"Randomized Kaczmarz methods with beyond-Krylov convergence."* arXiv:2501.11673, 2025.

[RFY$^+$25] Rathore, Frangella, Yang, Dereziński, and Udell. *"Have ASkotch: Fast cocktails for large-scale kernel ridge regression."* arXiv:2407.10070, 2025.

Case study: Large-scale Kernel Ridge Regression



Task: Fitting non-linear functions $f : \mathcal{X} \to \mathbb{R}$:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \Big(f(\phi_i) - y_i \Big)^2 + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

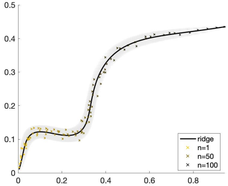
When $\mathcal{H}$ is a reproducing kernel Hilbert space defined by $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, this reduces to solving an $n \times n$ linear system:

$$\underbrace{(\mathbf{K} + n\lambda \mathbf{I})}_{\text{low-rank structure}} \mathbf{x} \;=\; \mathbf{y}, \qquad \text{where} \quad \mathbf{K} = \big[k(\phi_i, \phi_j)\big]_{i,j}.$$

Case study: Large-scale Kernel Ridge Regression

Task: Fitting non-linear functions $f : \mathcal{X} \to \mathbb{R}$:

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \Big(f(\phi_i) - y_i \Big)^2 + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$



When $\mathcal{H}$ is a reproducing kernel Hilbert space defined by $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, this reduces to solving an $n \times n$ linear system:

$$\underbrace{(\mathbf{K} + n\lambda\mathbf{I})}_{\text{low-rank structure}} \mathbf{x} = \mathbf{y}, \qquad \text{where} \quad \mathbf{K} = \big[k(\phi_i, \phi_j)\big]_{i,j}.$$

What if we are given 100 *million data points* (ϕ_i, y_i)*?*

Solving a dense $10^8 \times 10^8$ linear system

Solving KRR for New York City taxi transportation data ($n = 10^8$)

1. Storage: $40,000$TB (terabytes) to store $\mathbf{K}$ in single precision
2. Compute: State-of-the-art solvers take $>$ 24h for single iteration

Solving a dense $10^8 \times 10^8$ linear system

Solving KRR for New York City taxi transportation data ($n = 10^8$)

1. Storage: $40,000$TB (terabytes) to store $\mathbf{K}$ in single precision
2. Compute: State-of-the-art solvers take $>$ 24h for single iteration

Popular workarounds:

- Compress the matrix $\mathbf{K}$ and solve a smaller problem
- SGD-type solvers with heuristically chosen hyper-parameters

Solving a dense $10^8 \times 10^8$ linear system

Solving KRR for New York City taxi transportation data ($n = 10^8$)

1. Storage: $40,000$TB (terabytes) to store $\mathbf{K}$ in single precision
2. Compute: State-of-the-art solvers take $>$ 24h for single iteration

Popular workarounds:

- Compress the matrix $\mathbf{K}$ and solve a smaller problem
- SGD-type solvers with heuristically chosen hyper-parameters

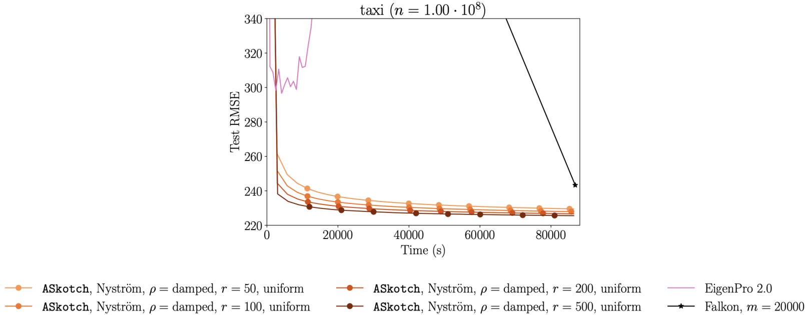
We attack the original problem with a provably convergent solver!

Solving a dense $10^8 \times 10^8$ linear system

Method: ASkotch ("Sketch-and-Project++" developed for GPUs)

Baselines: EigenPro 2.0 [MB19] and Falkon [RCR17]

Test RMSE: Root mean squared error on the test set.



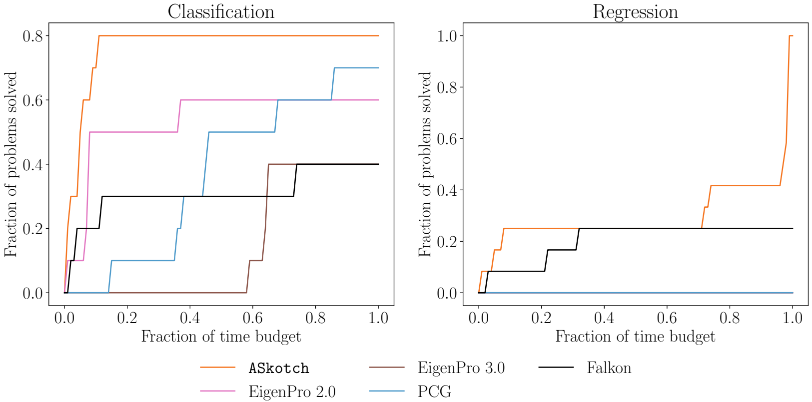
[RFY+25] Rathore, Frangella, Yang, Dereziński, and Udell. *"Have ASkotch: Fast cocktails for large-scale kernel ridge regression."* arXiv:2407.10070, 2025.

Case study: Large-scale Kernel Ridge Regression

Method: ASkotch ("Sketch-and-Project++" developed for GPUs)

Baselines: EigenPro [MB19], Falkon [RCR17], and Nyström PCG [FTU23]

Experiment: 23 tasks, including particle physics (4 datasets) and computational chemistry (9 datasets), with dimensions at least 10^5.



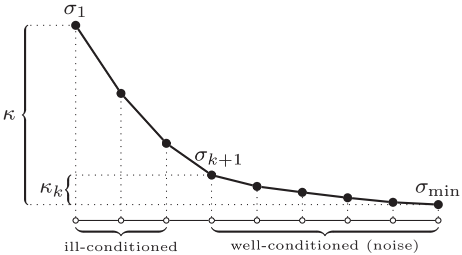
The code is available at `https://github.com/pratikrathore8/fast_krr`.

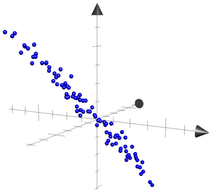
Outline

Roadmap

Complexity of solving linear systems with low-rank structure:

$$\text{Krylov methods:} \qquad \tilde{O}\left(n^2 k \;+\; n^2 \kappa_k \right)$$



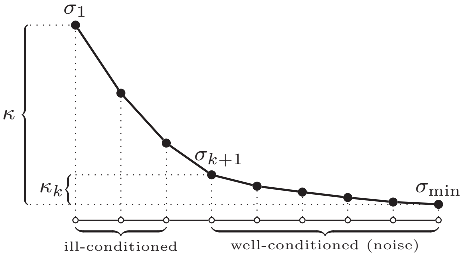


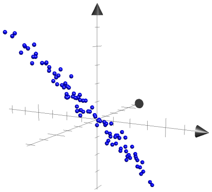
Roadmap

Complexity of solving linear systems with low-rank structure:

Krylov methods: $\tilde{O}\left(n^2 k \; + \; n^2 \kappa_k \right)$

Sketch-and-Project++: $\tilde{O}\left(n k^2 \; + \; n^2 \kappa_k \right)$





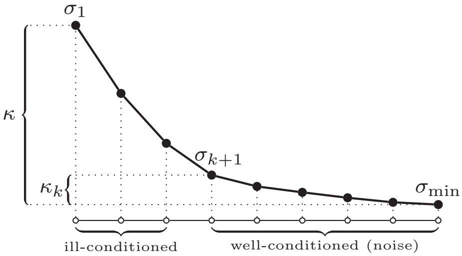
Roadmap

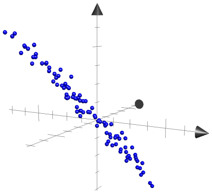
Complexity of solving linear systems with low-rank structure:

Krylov methods: $\tilde{O}\left(n^2 k + n^2 \kappa_k \right)$

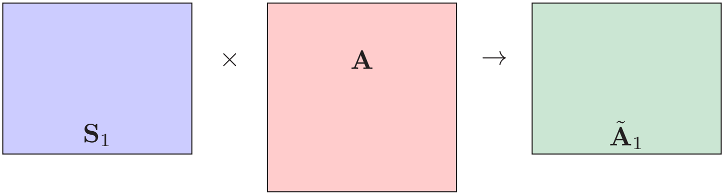
Sketch-and-Project++: $\tilde{O}\left(n k^2 + n^2 \kappa_k \right)$

Recursive Sketching: $\tilde{O}\left(k^3 + n^2 \kappa_k \right)$





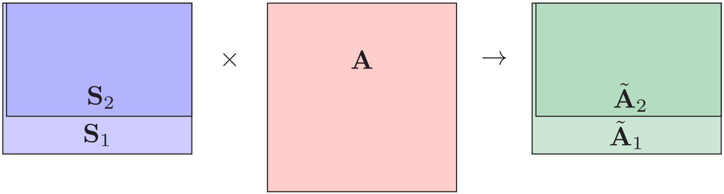
Key idea: Recursive Sketching



1. Precondition $\mathbf{A}$ using large sketch $\tilde{\mathbf{A}}_1 = \mathbf{S}_1\mathbf{A}$,

[DS25] Dereziński and Sidford. *"Approaching Optimality for Solving Dense Linear Systems with Low-Rank Structure."* arXiv:2507.11724, 2025.

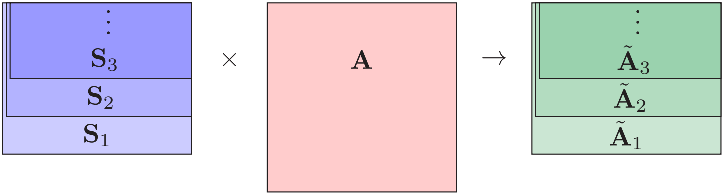
Key idea: Recursive Sketching



1. Precondition $\mathbf{A}$ using large sketch $\tilde{\mathbf{A}}_1 = \mathbf{S}_1\mathbf{A}$,
2. Precondition $\tilde{\mathbf{A}}_1$ using smaller sketch $\tilde{\mathbf{A}}_2 = \mathbf{S}_2\mathbf{A}$,

[DS25] Dereziński and Sidford. *"Approaching Optimality for Solving Dense Linear Systems with Low-Rank Structure."* arXiv:2507.11724, 2025.

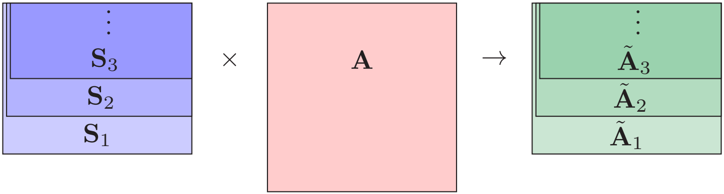
Key idea: Recursive Sketching



1. Precondition $\mathbf{A}$ using large sketch $\tilde{\mathbf{A}}_1 = \mathbf{S}_1\mathbf{A}$,
2. Precondition $\tilde{\mathbf{A}}_1$ using smaller sketch $\tilde{\mathbf{A}}_2 = \mathbf{S}_2\mathbf{A}$,
3. Precondition $\tilde{\mathbf{A}}_2$ using even smaller sketch $\tilde{\mathbf{A}}_3 = \mathbf{S}_3\mathbf{A}$, etc.

[DS25] Dereziński and Sidford. *"Approaching Optimality for Solving Dense Linear Systems with Low-Rank Structure."* arXiv:2507.11724, 2025.

Key idea: Recursive Sketching



1. Precondition $\mathbf{A}$ using large sketch $\tilde{\mathbf{A}}_1 = \mathbf{S}_1\mathbf{A}$,
2. Precondition $\tilde{\mathbf{A}}_1$ using smaller sketch $\tilde{\mathbf{A}}_2 = \mathbf{S}_2\mathbf{A}$,
3. Precondition $\tilde{\mathbf{A}}_2$ using even smaller sketch $\tilde{\mathbf{A}}_3 = \mathbf{S}_3\mathbf{A}$, etc.

Inspired by domain-specific preconditioning techniques:

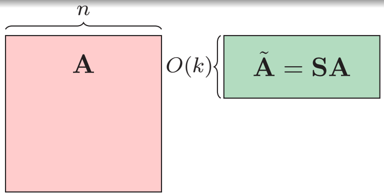
- Recursive solvers for graph Laplacians in theoretical computer science
- Multigrid solvers for differential equations in scientific computing

[DS25] Dereziński and Sidford. *"Approaching Optimality for Solving Dense Linear Systems with Low-Rank Structure."* arXiv:2507.11724, 2025.

Reminder: Single-sketch preconditioner fails

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, \ldots, k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



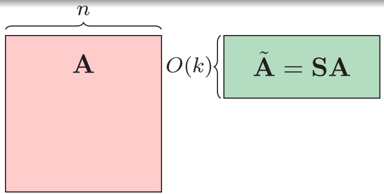
Bottom line: The sketch picks up **noise** from the **spectral tail**

$$\Rightarrow \qquad \tilde{\mathbf{A}}^\top\tilde{\mathbf{A}} \not\approx \mathbf{A}^\top\mathbf{A}$$

Reminder: Single-sketch preconditioner fails

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, \ldots, k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



Bottom line: The sketch picks up **noise** from the **spectral tail**

$$\Rightarrow \qquad \tilde{\mathbf{A}}^\top\tilde{\mathbf{A}} \not\approx \mathbf{A}^\top\mathbf{A}$$

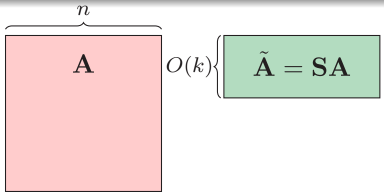
$$\text{but} \qquad \tilde{\mathbf{A}}^\top\tilde{\mathbf{A}} + \lambda\mathbf{I} \approx \underbrace{\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I}}_{\text{more low-rank}}, \qquad \text{for } \lambda = \frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A}).$$

Idea: Impose stricter low-rank structure on the system.

Reminder: Single-sketch preconditioner fails

Fact. In time $\tilde{O}(n^2)$, we can compute $\tilde{\mathbf{A}} \in \mathbb{R}^{O(k)\times n}$ such that for $i = 1, ..., k$:

$$\sigma_i^2(\tilde{\mathbf{A}}) = (1 \pm \tfrac{1}{2})\sigma_i^2(\mathbf{A}) \pm \underbrace{\frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A})}_{\text{tail noise}}$$



Bottom line: The sketch picks up **noise** from the **spectral tail**

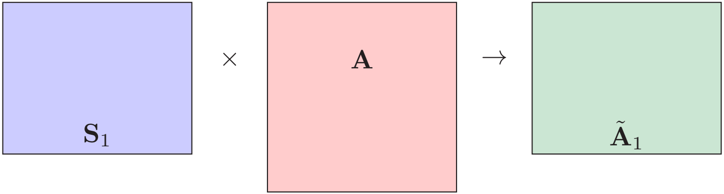
$$\Rightarrow \qquad \tilde{\mathbf{A}}^\top\tilde{\mathbf{A}} \not\approx \mathbf{A}^\top\mathbf{A}$$

$$\text{but} \qquad \tilde{\mathbf{A}}^\top\tilde{\mathbf{A}} + \lambda\mathbf{I} \approx \underbrace{\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I}}_{\text{more low-rank}}, \qquad \text{for } \lambda = \frac{1}{k}\sum_{j>k}\sigma_j^2(\mathbf{A}).$$

Idea: Impose stricter low-rank structure on the system.

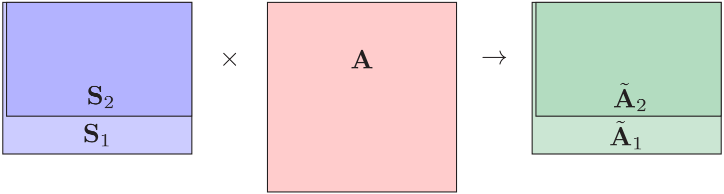
Wait! Are we allowed to do that?

Strategy: Gradually impose stricter low-rank structure



$$
\begin{aligned}
\mathbf{A}^\top\mathbf{A} \;\approx\; & \mathbf{A}^\top\mathbf{A} + \sigma_n^2\mathbf{I} \\
& \wr\wr \\
& \tilde{\mathbf{A}}_1^\top\tilde{\mathbf{A}}_1 + \lambda_1\mathbf{I}, \qquad \lambda_1 > \sigma_n^2
\end{aligned}
$$

Strategy: Gradually impose stricter low-rank structure



$$\mathbf{A}^\top \mathbf{A} \;\approx\; \mathbf{A}^\top \mathbf{A} + \sigma_n^2 \mathbf{I}$$

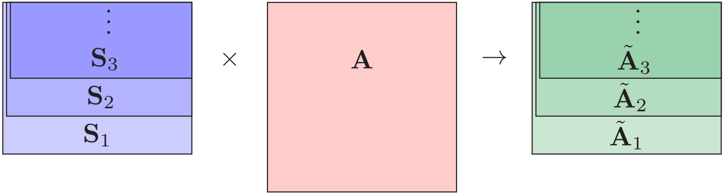
$$\wr\wr$$

$$\tilde{\mathbf{A}}_1^\top \tilde{\mathbf{A}}_1 + \lambda_1 \mathbf{I}, \qquad \lambda_1 > \sigma_n^2$$

$$\wr\wr$$

$$\tilde{\mathbf{A}}_2^\top \tilde{\mathbf{A}}_2 + \lambda_2 \mathbf{I}, \qquad \lambda_2 > \lambda_1$$

Strategy: Gradually impose stricter low-rank structure



$$
\begin{array}{llll}
\mathbf{A}^\top\mathbf{A} & \approx & \mathbf{A}^\top\mathbf{A} + \sigma_n^2\mathbf{I} & \\
 & & \wr\wr & \\
 & & \tilde{\mathbf{A}}_1^\top\tilde{\mathbf{A}}_1 + \lambda_1\mathbf{I}, & \lambda_1 > \sigma_n^2 \\
 & & \wr\wr & \\
 & & \tilde{\mathbf{A}}_2^\top\tilde{\mathbf{A}}_2 + \lambda_2\mathbf{I}, & \lambda_2 > \lambda_1 \\
 & & \wr\wr & \\
 & & \tilde{\mathbf{A}}_3^\top\tilde{\mathbf{A}}_3 + \lambda_3\mathbf{I}, & \lambda_3 > \lambda_2 \\
 & & \vdots &
\end{array}
$$

Recursive Sketching: Main result

Theorem ([DS25])

Given any k, we can solve an $O(n) \times n$ linear system $\mathbf{Ax} = \mathbf{b}$ in time:

$$\tilde{O}\big(k^3 + n^2 \kappa_k \log 1/\epsilon\big), \quad \textit{where} \quad \kappa_k = \tfrac{\sigma_{k+1}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}.$$

- Natural complexity limit for systems with low-rank structure
- Even better: κ_k can be replaced by a *smoothed* condition number,

$$\bar{\kappa}_k = \frac{1}{n-k} \sum_{i>k} \frac{\sigma_i}{\sigma_{\min}} < \kappa_k$$

Application: The first algorithm for approximating the nuclear norm $\|\mathbf{A}\|_1 = \sum_i \sigma_i$ of an $n \times n$ matrix in nearly linear time $\tilde{O}(n^2)$.

[DS25] Dereziński and Sidford. *"Approaching Optimality for Solving Dense Linear Systems with Low-Rank Structure."* arXiv:2507.11724, 2025.

Outline

Conclusions

Linear systems with low-rank structure: a natural model for ill-conditioned matrices, e.g., in ML/Opt/Stats applications.

Randomized algorithms:

- *Sketch-and-Project++*: New family of randomized linear solvers that scaled to massive problem sizes.
- *Recursive Sketching*: Attains nearly optimal complexity for solving dense linear systems with low-rank structure.

Next steps:

- *Extending to sparse matrices and other access models*
- *Extending to other natural singular/eigenvalue profiles*

References I

Owe Axelsson and Gunhild Lindskog.
On the rate of convergence of the preconditioned conjugate gradient method.
Numerische Mathematik, 48:499–523, 1986.

Haim Avron, Petar Maymounkov, and Sivan Toledo.
Blendenpik: Supercharging lapack's least-squares solver.
SIAM Journal on Scientific Computing, 32(3):1217–1236, 2010.

Michał Dereziński, Feynman T Liang, Zhenyu Liao, and Michael W Mahoney.
Precise expressions for random projections: Low-rank approximation and randomized newton.
Advances in Neural Information Processing Systems, 33, 2020.

Michał Dereziński, Daniel LeJeune, Deanna Needell, and Elizaveta Rebrova.
Fine-grained analysis and faster algorithms for iteratively solving linear systems.
arXiv preprint arXiv:2405.05818, 2024.

Michał Dereziński, Deanna Needell, Elizaveta Rebrova, and Jiaming Yang.
Randomized kaczmarz methods with beyond-krylov convergence.
arXiv preprint arXiv:2501.11673, 2025.

Michał Dereziński and Elizaveta Rebrova.
Sharp analysis of sketch-and-project methods via a connection to randomized singular value decomposition.
SIAM Journal on Mathematics of Data Science, 6(1):127–153, 2024.

Michał Dereziński and Aaron Sidford.
Approaching optimality for solving dense linear systems with low-rank structure.
arXiv preprint arXiv:2507.11724, 2025.

References II

Michał Dereziński and Jiaming Yang.
Solving dense linear systems faster than via preconditioning.
In 56th Annual ACM Symposium on Theory of Computing, 2024.

Zachary Frangella, Joel A Tropp, and Madeleine Udell.
Randomized Nyström preconditioning.
SIAM Journal on Matrix Analysis and Applications, 44(2):718–752, 2023.

Robert M Gower and Peter Richtárik.
Randomized iterative methods for linear systems.
SIAM Journal on Matrix Analysis and Applications, 36(4):1660–1690, 2015.

M. S. Kaczmarz.
Angenaherte auflosung von systemen linearer gleichungen.
Bulletin International de l'Academie Polonaise des Sciences et des Lettres, 35:355–357, 1937.

Siyuan Ma and Mikhail Belkin.
Kernel machines that adapt to gpus for effective large batch training.
Proceedings of Machine Learning and Systems, 1:360–373, 2019.

Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco.
Falkon: An optimal large scale kernel method.
Advances in neural information processing systems, 30, 2017.

Pratik Rathore, Zachary Frangella, Jiaming Yang, Michał Dereziński, and Madeleine Udell.
Have askotch: A neat solution for large-scale kernel ridge regression.
arXiv preprint arXiv:2407.10070, 2025.

References III

Thomas Strohmer and Roman Vershynin.
A randomized kaczmarz algorithm with exponential convergence.
Journal of Fourier Analysis and Applications, 15(2):262–278, 2009.

Pseudocode for Kaczmarz++

back to slide

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, B, k, $\mathbf{x}_0$, $t_{\max}$, η, ρ, λ

$\mathbf{D} \rightarrow \mathrm{diag}(\text{random } \pm 1/\sqrt{m})$

$\mathbf{A} \leftarrow \mathbf{HDA}$ and $\mathbf{b} \leftarrow \mathbf{HDb}$ ▷ Randomized Hadamard

$\mathbf{m}_0 \leftarrow \mathbf{0}$;

Sample $\mathcal{B} = \{S_1, ..., S_B\} \subseteq \binom{[m]}{k}$ ▷ Random blocks

for $t = 0, 1, \ldots, (t_{\max} - 1)$ **do**

  Draw a random subset S from $\mathcal{B}$ ▷ Fast sketching

  $\mathbf{r}_t \leftarrow \mathbf{A}_S \mathbf{x}_t - \mathbf{b}_S$

  $\mathbf{w}_t \approx \mathrm{argmin}_{\mathbf{w}} \left\{ \|\mathbf{A}_S \mathbf{w} - \mathbf{r}_t\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$ ▷ Fast projection

  $\mathbf{m}_{t+1} \leftarrow \frac{1-\rho}{1+\rho}(\mathbf{m}_t - \mathbf{w}_t)$

  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \mathbf{w}_t + \eta \mathbf{m}_t$ ▷ Momentum acceleration

end for

return $\tilde{\mathbf{x}} = \mathbf{x}_{t_{\max}}$;

The code is available at `https://github.com/EdwinYang7/kaczmarz-plusplus`