Randomized algorithms for very large scale linear algebra

Flatiron Institute, May 6, 2026

Gunnar Martinsson
University of Texas at Austin

Slides & surveys at: https://users.oden.utexas.edu/∼pgm/

Across computational science, a few linear algebraic tasks reappear

- Solving linear systems.

  Discretized PDEs, data fitting, implicit methods, . . .

- Eigenvalue problems.

  Computational chemistry, materials science, wave problems, buckling, PageRank, . . .

- Low rank approximation.

  Model reduction, data and image compression, principal component analysis, . . .

The execution speed of these computational kernels often determines what scale, resolution, or accuracy is computationally feasible.

Why look for new methods? What changed?

Textbook methods for linear algebraic computations were designed for an environment where the main concern was *arithmetic cost*.

On modern hardware, different costs often dominate:

- moving data between levels of memory,
- moving data across processors,
- making repeated passes over a massive matrix.

Randomized methods often require much less data movement.

Basic algorithmic template

- Take a matrix or operator whose size, or repeated appearance, makes a computation expensive.
- Probe it by observing its action on a small number of randomized test vectors.
  These probes can often be extracted in parallel.
- Recover the parts that matter, and discard components that got caught accidentally.

$$\textit{Probe} \to \textit{Recover structure} \to \textit{Compute on the compressed object}$$

In many important cases, this yields:

- high accuracy,
- major speedups,
- much less communication,
- access to problems that were previously out of reach.

Three case studies

1. **Low-rank approximation & eigenvalue problems**

   Compressing information while preserving accuracy.

2. **Sketching for least squares and linear systems**

   Solving equations faster. (Some equations, at least...)

3. **Global operators from mathematical physics**

   Compressing dense operators so that we can compute with them.

Part 1: Low rank approximation

Matrices and tensors abound in scientific computing, data science, imaging, …

Redundancy is common – data can be described well by its dominant modes.

- principal component analysis,
- model reduction in simulation,
- spectral methods in data analysis,
- fast algorithms for structured matrices.

The first part of the talk is about exploiting this redundancy.

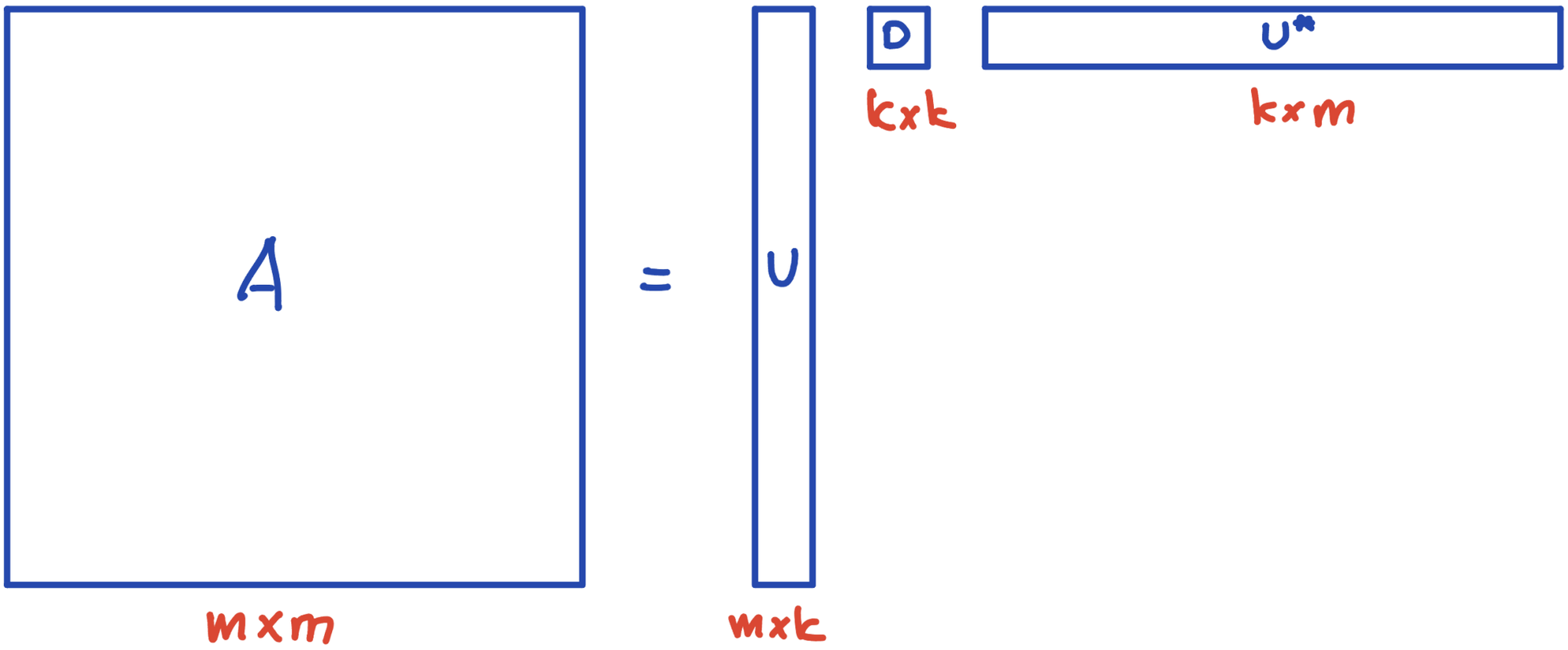
Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

For instance

$$\underset{m\times m}{\mathbf{A}} \approx \underset{m\times k}{\mathbf{U}}\ \underset{k\times k}{\mathbf{D}}\ \underset{k\times m}{\mathbf{U}^*}.$$



Retain the essential action of **A** using far fewer numbers.

Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Classical answers:

- Gram-Schmidt on the columns of **A**.
- Power iteration to find eigenvectors.
- Krylov methods.
- …

Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that **A** has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\mathbf{\Omega}$ and form the product $\mathbf{A\Omega}$. Then, with probability 1,

(NYS)
$$\underset{m \times m}{\mathbf{A}} = \underset{m \times k}{(\mathbf{A\Omega})} \ \underset{k \times k}{(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}} \ \underset{k \times m}{(\mathbf{A\Omega})^*}.$$

Low rank approximation:

Let $\mathbf{A}$ be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that $\mathbf{A}$ has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\mathbf{\Omega}$ and form the product $\mathbf{A\Omega}$. Then, with probability 1,

(NYS)
$$\underset{m\times m}{\mathbf{A}} = \underset{m\times k}{(\mathbf{A\Omega})}\ \underset{k\times k}{(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}}\ \underset{k\times m}{(\mathbf{A\Omega})^*}.$$

Proof: Let the exact eigenvalue decomposition of $\mathbf{A}$ be $\mathbf{A} = \mathbf{UDU}^*$. Then

$$\mathbf{A\Omega} = \mathbf{UDU}^*\mathbf{\Omega} = \{\mathbf{\Psi} := \mathbf{U}^*\mathbf{\Omega}\} = \mathbf{UD\Psi},$$

where $\mathbf{\Psi}$ is a $k \times k$ matrix that also has a Gaussian distribution. Then

$$(\mathbf{A\Omega})(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}(\mathbf{A\Omega})^* = (\mathbf{UD\Psi})(\mathbf{\Psi}^*\mathbf{D\Psi})^{-1}(\mathbf{UD\Psi})^*$$

Observe that $\mathbf{\Psi}$ is invertible with probability 1, so

$$(\mathbf{A\Omega})(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}(\mathbf{A\Omega})^* = \mathbf{UD\Psi}\,\mathbf{\Psi}^{-1}\mathbf{D}^{-1}\mathbf{\Psi}^{-*}\mathbf{\Psi}^*\mathbf{DU}^* = \mathbf{UDU}^* = \mathbf{A}.$$

Low rank approximation:

Let $\mathbf{A}$ be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that $\mathbf{A}$ has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\boldsymbol{\Omega}$ and form the product $\mathbf{A\Omega}$. Then, with probability 1,

$$\underset{m \times m}{\mathbf{A}} = \underset{m \times k}{(\mathbf{A\Omega})} \; \underset{k \times k}{(\boldsymbol{\Omega}^*\mathbf{A\Omega})^{-1}} \; \underset{k \times m}{(\mathbf{A\Omega})^*}. \tag{NYS}$$

Exploiting (NYS), you can easily compute an exact eval decomposition $\mathbf{A} = \mathbf{UDU}^*$:

1. Draw an $m \times k$ Gaussian random matrix $\boldsymbol{\Omega}$. `Omega = randn(m,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\mathrm{col}(\mathbf{Y}) = \mathrm{col}(\mathbf{Q})$. `[Q, ~] = qr(Y,0)`
4. Form the $k \times k$ symmetric matrix: $\mathbf{B} = (\boldsymbol{\Omega}^*\mathbf{Q})^{-1}\mathbf{R}^*$. `B = inv(Omega'*Q)*R'`
5. Compute the EVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\hat{\mathbf{U}}^*$. `[Uhat, D] = eig(B)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

The cost is $O(m^2k)$ for a dense matrix. *Same as classical methods!*

Low rank approximation:

Let $\mathbf{A}$ be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that $\mathbf{A}$ has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\boldsymbol{\Omega}$ and form the product $\mathbf{A}\boldsymbol{\Omega}$. Then, with probability 1,

$$\text{(NYS)} \qquad \underset{m \times m}{\mathbf{A}} = \underset{m \times k}{(\mathbf{A}\boldsymbol{\Omega})} \ \underset{k \times k}{(\boldsymbol{\Omega}^*\mathbf{A}\boldsymbol{\Omega})^{-1}} \ \underset{k \times m}{(\mathbf{A}\boldsymbol{\Omega})^*}.$$

Exploiting (NYS), you can easily compute an exact eval decomposition $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^*$:

1. Draw an $m \times k$ Gaussian random matrix $\boldsymbol{\Omega}$. `Omega = randn(m,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\mathrm{col}(\mathbf{Y}) = \mathrm{col}(\mathbf{Q})$. `[Q, ~] = qr(Y,0)`
4. Forı `*Q)*R')`
5. Conı `eig(B)`
6. Forı `* Uhat`

> ☞ The interaction with $\mathbf{A}$ is only through the matrix-matrix product.
> Very high *wall clock* speed.
> Key reason for the practical impact of these methods.

The cost is $O(m^2k)$ for a dense matrix. *Same as classical methods!*

Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that **A** has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\mathbf{\Omega}$ and form the product $\mathbf{A\Omega}$. Then, with probability 1,

$$\underset{m \times m}{\mathbf{A}} = \underset{m \times k}{(\mathbf{A\Omega})} \ \underset{k \times k}{(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}} \ \underset{k \times m}{(\mathbf{A\Omega})^*}. \tag{NYS}$$

Observation:

The randomized algorithm described can be executed in a *streaming* mode.

You only need to see each entry of the matrix once!

The deterministic methods we described cannot do this.

Low rank approximation:

Let $\mathbf{A}$ be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Question: How do you compute the smaller factors efficiently?

Randomized algorithm:

Warm-up case: Suppose that $\mathbf{A}$ has *exact* rank k and is positive semi-definite.

Draw an $m \times k$ Gaussian matrix $\mathbf{\Omega}$ and form the product $\mathbf{A\Omega}$. Then, with probability 1,

$$\underset{m \times m}{\mathbf{A}} = \underset{m \times k}{(\mathbf{A\Omega})} \ \underset{k \times k}{(\mathbf{\Omega}^*\mathbf{A\Omega})^{-1}} \ \underset{k \times m}{(\mathbf{A\Omega})^*}. \tag{NYS}$$

Observation: We can improve on the asymptotic complexity.

The key is to use a *structured* random matrix $\mathbf{\Omega}$ for which $\mathbf{A\Omega}$ can be evaluated rapidly.

- Randomly scrambled discrete Fourier transforms — apply via FFT.
- Chains of Givens rotations between pairs of columns. "Kac's random walk."
- The matrix $\mathbf{\Omega}$ can even be *sparse*. Need ≥ 2 non-zeros per row.

Reduction in asymptotic complexity from $O(m^2 k)$ to $O(m^2 + mk^2)$.

Low rank approximation:

Let **A** be a large matrix of numerically low rank. Say symmetric.

We seek to compute a low rank factorization.

Questic

Randor

Warm-u e.

Draw ar ability 1,

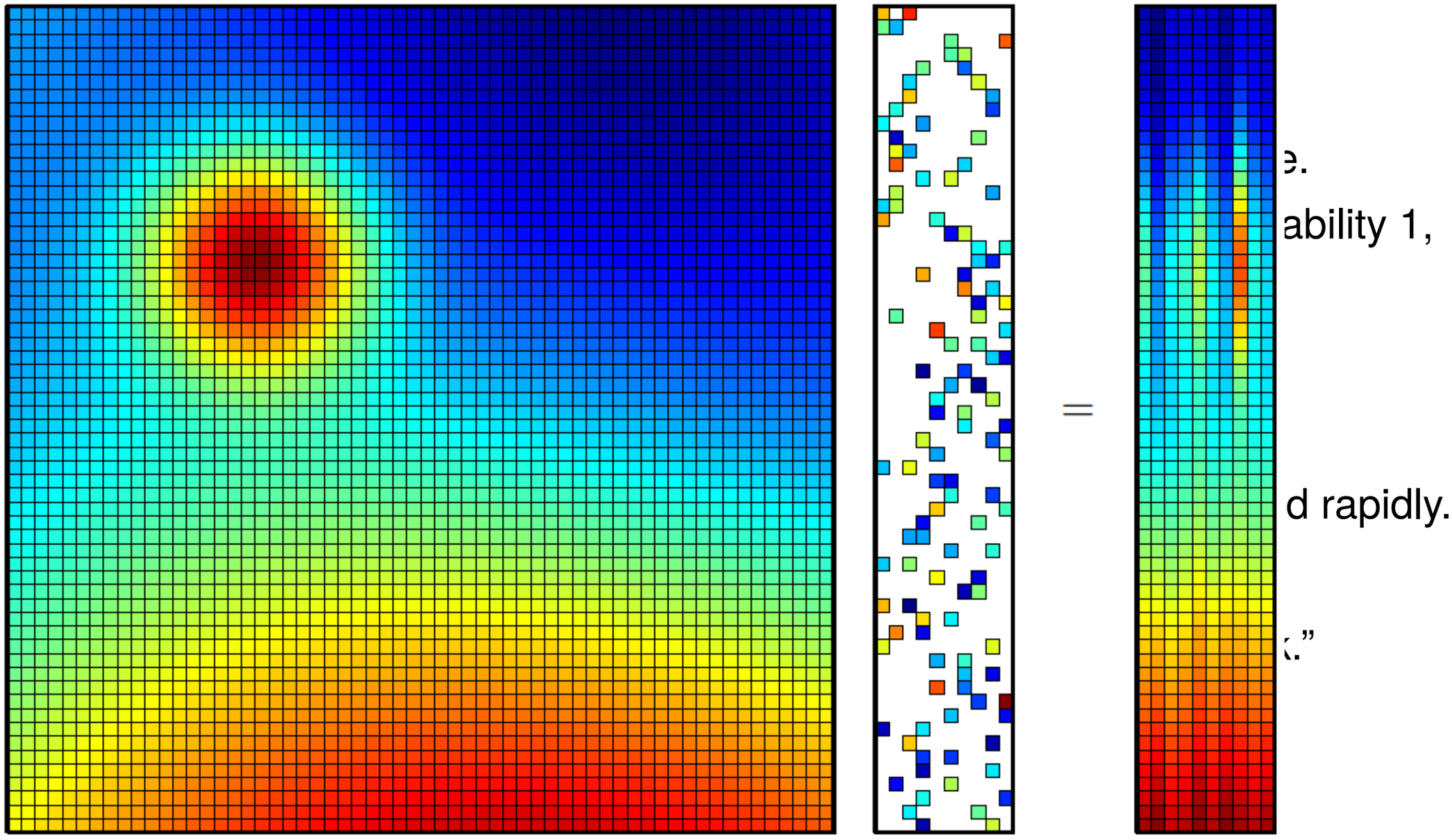
(NYS)

Observ

The key d rapidly.

- Rar
- Cha ."
- The

Reducti



Low rank approximation:

Next, let us widen the scope.

Warm-up case: **A** is $m \times m$ symmetric, non-negative, exact rank k, seek EVD:

$$\begin{array}{ccccc} \mathbf{A} & = & \mathbf{U} & \mathbf{D} & \mathbf{U}^*. \\ m \times m & & m \times k & k \times k & k \times m \end{array}$$

General case: **A** is $m \times n$, approximate rank k, we seek singular value decomposition:

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{U} & \mathbf{D} & \mathbf{V}^*. \\ m \times n & & m \times k & k \times k & k \times n \end{array}$$

Where **U** and **V** are orthonormal, and **D** is diagonal.

The key features of the algorithm remain:

- Draw a thin random matrix $\mathbf{\Omega}$.
- Build a sample matrix $\mathbf{A\Omega}$ whose columns approximately span ran($\mathbf{A}$).
- Interaction with **A** is only through matrix-matrix multiplication.
- Streaming mode ("single-view") is still possible.
- Structured $\mathbf{\Omega}$ leads to acceleration from $O(mnk)$ to $O(mn + (m+n)k^2)$.

Low rank approximation:

Problem: Given an $m \times n$ matrix $\mathbf{A}$, and a target rank k, where $k \ll \min(m,n)$, we seek to compute an approximate partial singular value decomposition:

$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{U}} \; \underset{k \times k}{\mathbf{D}} \; \underset{k \times n}{\mathbf{V}^*},$$

with $\mathbf{U}$ and $\mathbf{V}$ having orthonormal columns, and $\mathbf{D}$ diagonal.

The "Randomized SVD (RSVD)":

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\text{col}(\mathbf{Y}) = \text{col}(\mathbf{Q})$. `[Q, ~] = qr(Y,0)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Power iteration: When the singular values of $\mathbf{A}$ decay slowly, precision can be improved by replacing the formula $\mathbf{Y} = \mathbf{A\Omega}$ on line 2 by $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*\mathbf{\Omega})$, or $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*(\mathbf{A\Omega}))$, or ...

Low rank approximation:

Problem: Given an $m \times n$ matrix $\mathbf{A}$, and a target rank k, where $k \ll \min(m,n)$, we seek to compute an approximate partial singular value decomposition:

$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{U}} \; \underset{k \times k}{\mathbf{D}} \; \underset{k \times n}{\mathbf{V}^*},$$

with $\mathbf{U}$ and $\mathbf{V}$ having orthonormal columns, and $\mathbf{D}$ diagonal.

The "Randomized SVD (RSVD)":

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ s. t. $\text{col}(\mathbf{Y}) = \text{col}(\mathbf{Q})$. `[Q, ~] = qr(Y,0)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of $\mathbf{B}$ (small!): $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

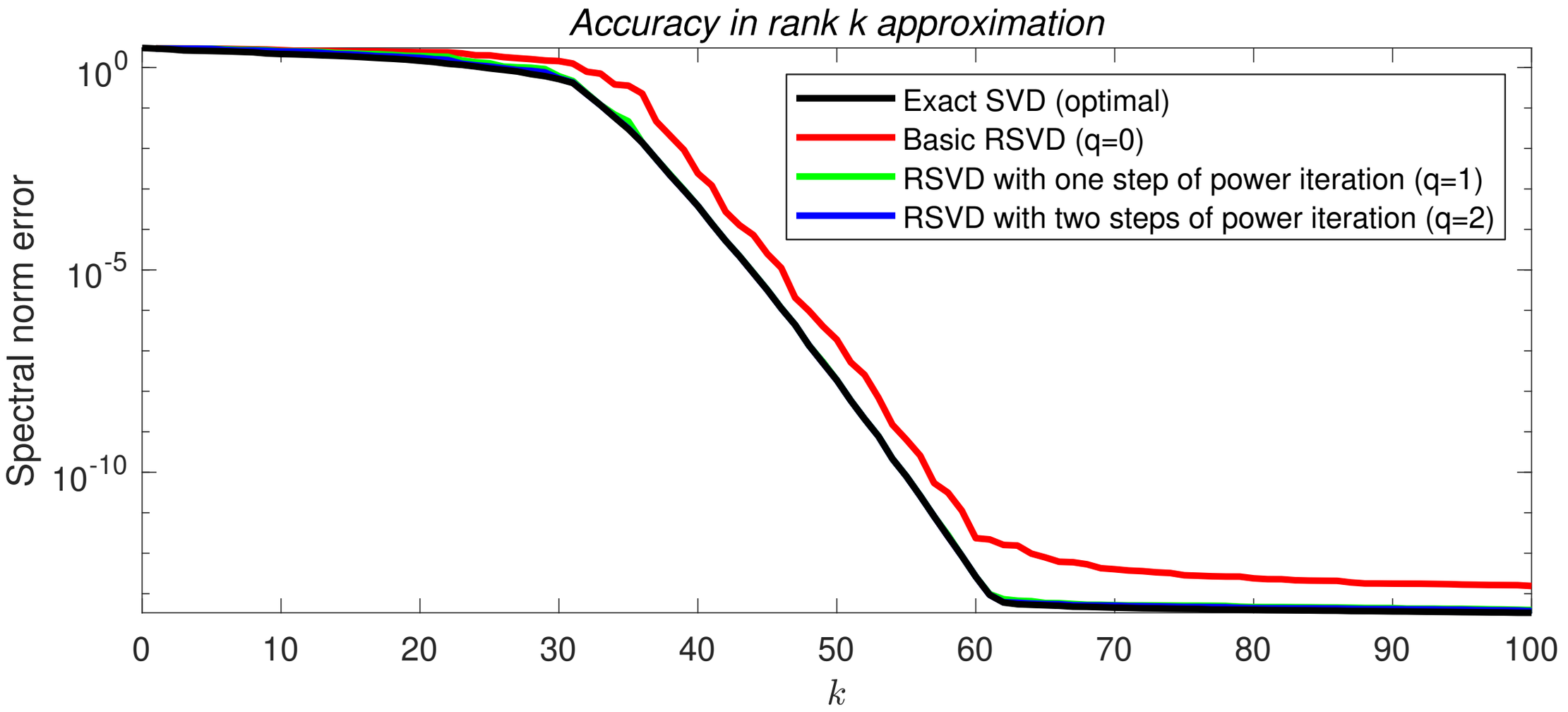
Error analysis: Steps (3) – (6) are exact, so

$$\mathbf{A}_{\text{approx}} := \mathbf{U}\mathbf{D}\mathbf{V}^* = \mathbf{Q}\hat{\mathbf{U}}\mathbf{D}\mathbf{V}^* = \mathbf{Q}\mathbf{B} = \mathbf{Q}\mathbf{Q}^*\mathbf{A} = \mathbf{P}_\mathbf{Y}\mathbf{A},$$

where $\mathbf{P}_\mathbf{Y}$ is the orthogonal projection onto $\text{span}(\mathbf{Y})$.

So the question is: How well do the columns of $\mathbf{Y} = \mathbf{A\Omega}$ span the columns of $\mathbf{A}$?

Low rank approximation:



The plot shows the errors from the randomized range finder. To be precise, we plot

$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

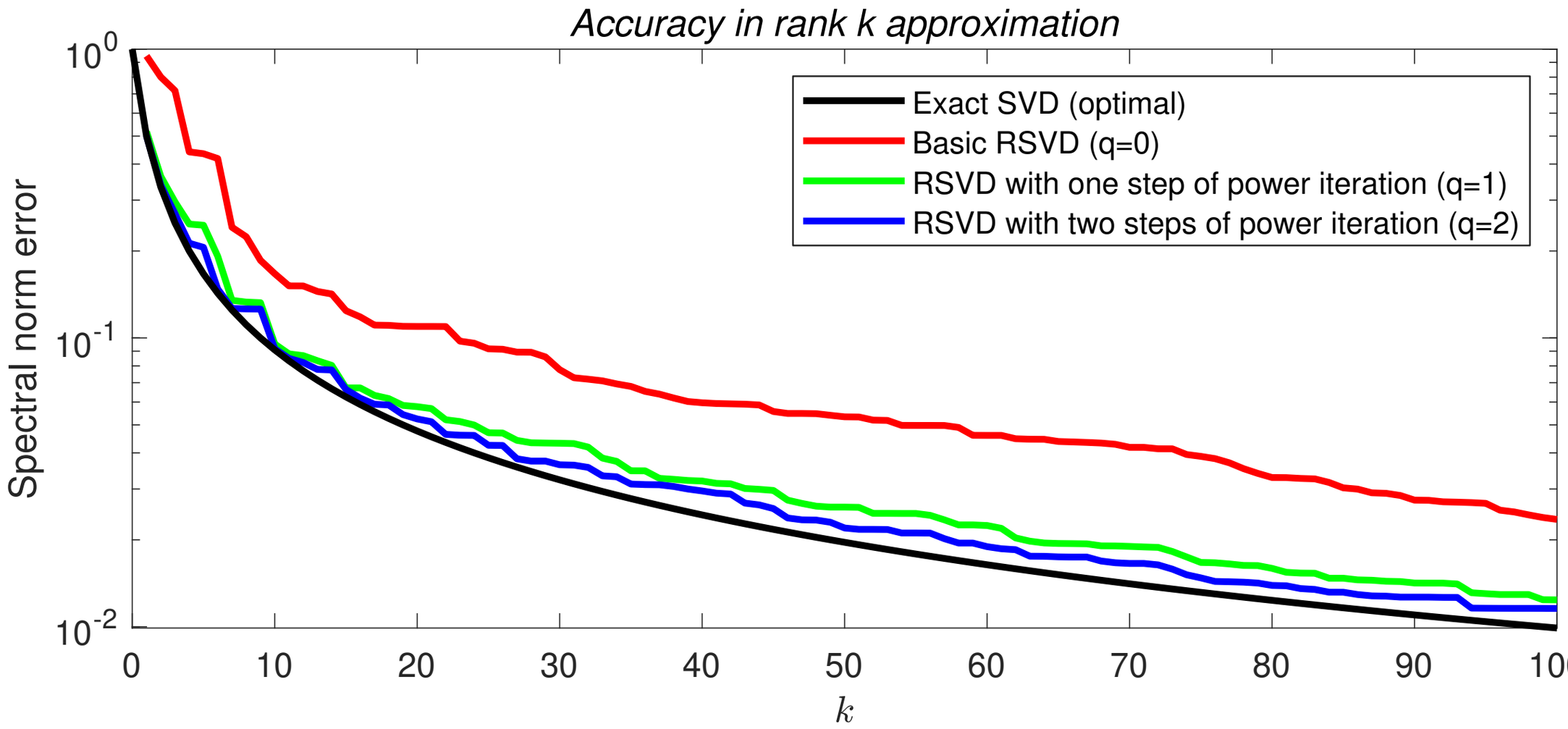
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

$$\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)^q\mathbf{A}\mathbf{\Omega},$$

and where $\mathbf{\Omega}$ is a Gaussian random matrix.

The matrix $\mathbf{A}$ is an approximation to a scattering operator for a Helmholtz problem.

Low rank approximation:



The plot shows the errors from the randomized range finder. To be precise, we plot

$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

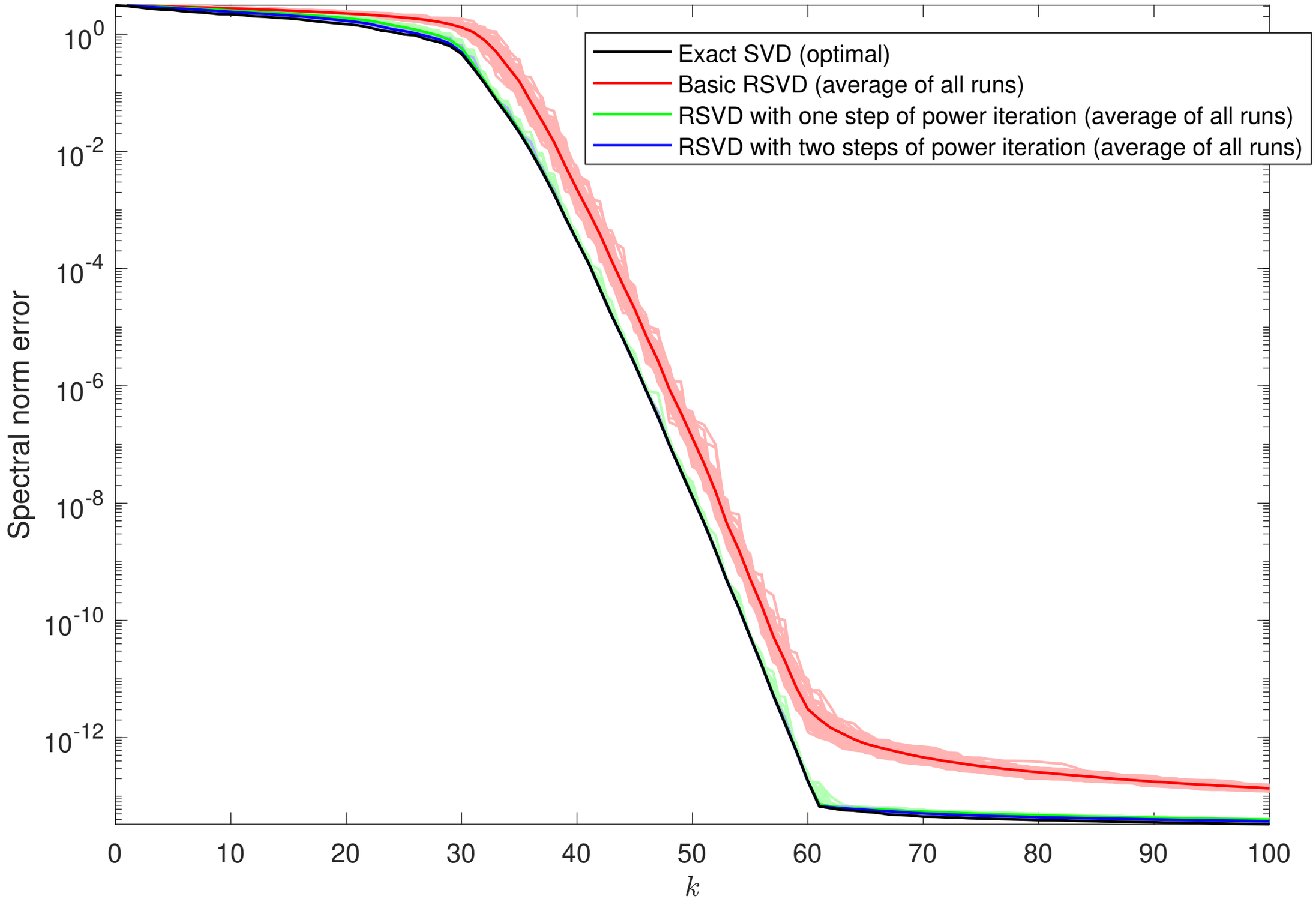
$$\mathbf{Y} = \left(\mathbf{A}\mathbf{A}^*\right)^q \mathbf{A}\mathbf{\Omega},$$

and where $\mathbf{\Omega}$ is a Gaussian random matrix.

The matrix $\mathbf{A}$ now has singular values that decay slowly.

Low rank approximation:

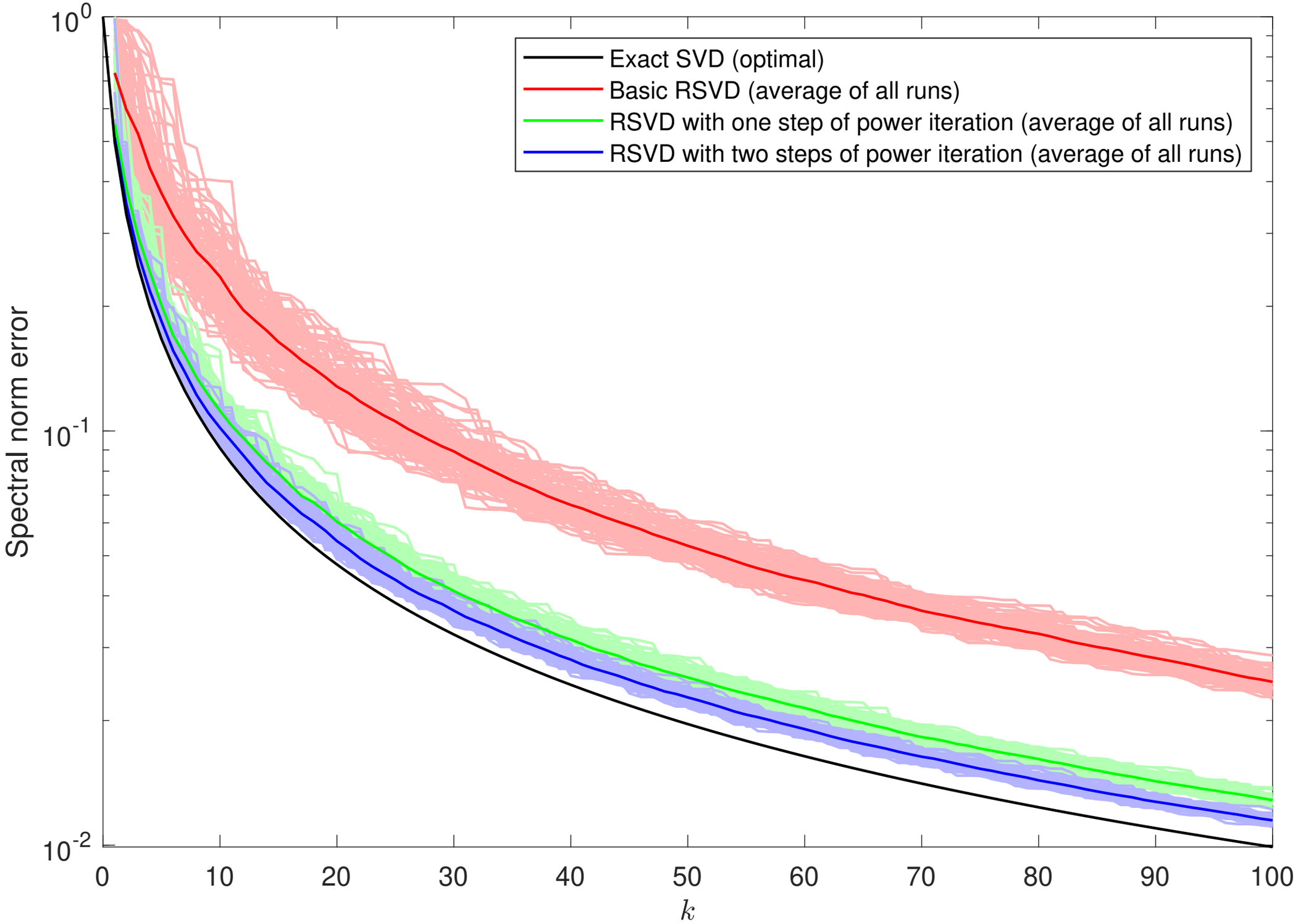
The same plot, but showing 100 instantiations.



The darker lines show the mean errors across the 100 experiments.

Low rank approximation:

The same plot, but showing 100 instantiations.



The darker lines show the mean errors across the 100 experiments.

Input: An $m \times n$ matrix $\mathbf{A}$ and a target rank k.	
Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times k$ random matrix $\mathbf{\Omega}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Error analysis:

Let $\mathbf{A} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{V}_\star^*$ be the exact singular value decomposition of $\mathbf{A}$. Recall that

$$\mathbf{A}_{\text{approx}} := \mathbf{UDV}^* = \mathbf{QQ}^*\mathbf{A} = \mathbf{P}_\mathbf{Y}\mathbf{A},$$

where $\mathbf{P}_\mathbf{Y}$ is the orthogonal projection onto $\text{span}(\mathbf{Y}) = \text{span}(\mathbf{A\Omega})$.

We can express the distribution of the columns of $\mathbf{Y}$ explicitly:

$$\mathbf{Y} = \mathbf{A\Omega} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{V}_\star^* \mathbf{\Omega} = \{\mathbf{\Psi} := \mathbf{V}_\star^* \mathbf{\Omega}\} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{\Psi}.$$

So the j'th column of $\mathbf{Y}$ is given by

$$\mathbf{y}^{(j)} = \sum_{i=1}^{\min(m,n)} \sigma_i \psi_{i,j} \, \mathbf{u}_\star^{(i)},$$

where $\{\sigma_i\}_{i=1}^{\min(m,n)}$ are the exact singular values of $\mathbf{A}$.

Power iteration: If we use instead $\mathbf{Y} = \mathbf{AA}^*\mathbf{A\Omega}$, then $\mathbf{y}^{(j)} = \sum_{i=1}^{\min(m,n)} \sigma_i^3 \psi_{i,j} \, \mathbf{u}_\star^{(i)}$.

Input: An $m \times n$ matrix $\mathbf{A}$ and a target rank k.	
Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times k$ random matrix $\mathbf{\Omega}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Error analysis:

Let $\mathbf{A} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{V}_\star^*$ be the exact singular value decomposition of $\mathbf{A}$. Recall that

$$\mathbf{A}_{\text{approx}} := \mathbf{UDV}^* = \mathbf{QQ}^*\mathbf{A} = \mathbf{P}_\mathbf{Y}\mathbf{A},$$

where $\mathbf{P}_\mathbf{Y}$ is the orthogonal projection onto $\text{span}(\mathbf{Y}) = \text{span}(\mathbf{A\Omega})$.

We can express the distribution of the columns of $\mathbf{Y}$ explicitly:

$$\mathbf{Y} = \mathbf{A\Omega} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{V}_\star^* \mathbf{\Omega} = \{\mathbf{\Psi} := \mathbf{V}_\star^* \mathbf{\Omega}\} = \mathbf{U}_\star \mathbf{D}_\star \mathbf{\Psi}.$$

So the j'th column of $\mathbf{Y}$ is given by

$$\mathbf{y}^{(j)} = \sum_{i=1}^{\min(m,n)} \sigma_i \psi_{i,j} \, \mathbf{u}_\star^{(i)},$$

where $\{\sigma_i\}_{i=1}^{\min(m,n)}$ are the exact singular values of $\mathbf{A}$.

Over-sampling is essential regardless of whether power iteration is used.

Theorem: (Halko, Martinsson, Tropp 2011) Let $\mathbf{A}$ be an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$. Let k be a target rank, and let p be an over-sampling parameter such that $p \geq 2$ and $k + p \leq \min(m, n)$. Let $\mathbf{\Omega}$ be a Gaussian random matrix of size $n \times (k + p)$ and let $\mathbf{P}_{\mathbf{A\Omega}}$ denote the orthogonal projection onto the range of $\mathbf{A\Omega}$. Then the average error satisfies

$$\mathrm{E}\big[\|\mathbf{A} - \mathbf{P}_{\mathbf{A\Omega}}\mathbf{A}\|_{\mathrm{Fro}}\big] \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2},$$

$$\mathbf{E}\big[\|\mathbf{A} - \mathbf{P}_{\mathbf{A\Omega}}\mathbf{A}\|\big] \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}.$$

Over-sampling is essential: At the core of the proof is a bound on the smallest singular value of a Gaussian random matrix of size $k \times (k + p)$ Gaussian matrix.

- With no oversampling, $p = 0$, we get terrible results!
- With lots of oversampling, $k \ll \ell$, things look good. ("Marchenko–Pastur" case)
- The interesting fact is that things stabilize very quickly. $p = 5$ is fine!

Over-sampling is also benign: Contributions from "unwanted" directions of the range are filtered out in the post-processing stage.

There are also bounds on the likelihood of a large deviation from the expectation.

Input: An $m \times n$ matrix **A**, a target rank k, and an over-sampling parameter p (say $p = 5$).	
Output: Rank-$(k+p)$ factors **U**, **D**, and **V** in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix **Q** s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key points:

- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Power iteration. Replace $\mathbf{Y} = \mathbf{A\Omega}$ by $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*\mathbf{\Omega})$, or $\mathbf{Y} = \mathbf{A}(\mathbf{A}^*(\mathbf{A\Omega}))$, or ...
- Communication efficient: Out-of-core, GPU, distributed memory, ...
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn)$.

  The key is to use a *Fast Johnson-Lindenstrauss transform*.
  - Randomized trigonometric transforms (FFT, Hadamard, etc). Cost is $O(mn\log(k))$.
  - Chains of Given's rotations ("Kac's random walk"). Cost is $O(mn\log(k))$.
  - Sparse random matrices: As few as 2 non-zeros per row. Cost is $O(mn)$!
- Single pass algorithms have been developed for *streaming environments.*

  The idea is that you are allowed to observe each matrix element only once.

  You cannot store the matrix. *Not possible with deterministic methods!*

Part 2: Randomized algorithms for solving linear systems

Much work in recent years on finding randomized methods for solving linear systems.

For huge linear systems, or for systems where the coefficient matrix cannot be fully evaluated (Hessians, kernel matrices, ...), Monte Carlo style methods are popular. Quite different computational profile.

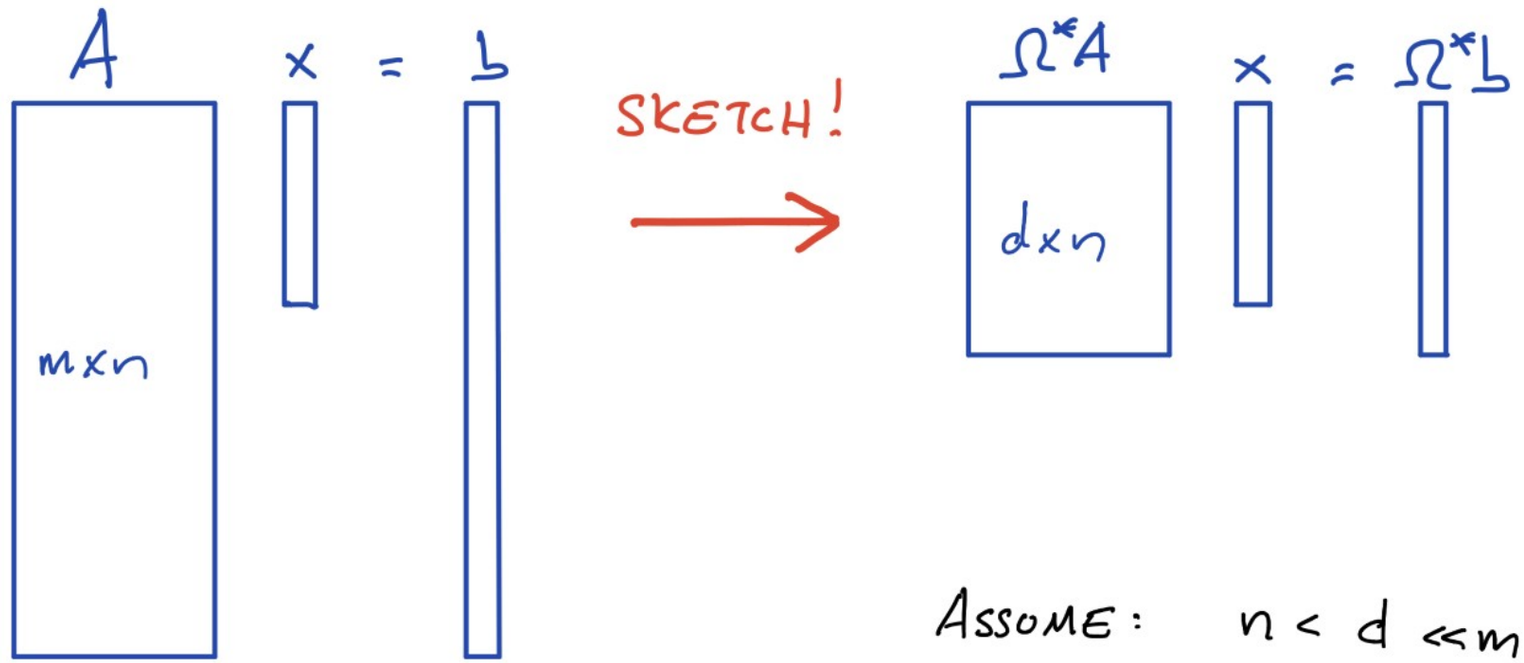
The "mixing methods" that we have described have had two key applications for solvers:

- Accelerate classical methods (Gaussian elimination, QR factorization, etc) by reducing *communication*. Asymptotic complexity remains $O(m^3)$.
- For over-determined least squares problems, we can probably reduce in complexity from $O(mnk)$ to $O(mn + k^3)$. *Topic of next couple of slides.*

Random mixing for solving linear systems & least squares problems

Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ for $m \gg n$, and that you seek to solve $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$.

Draw a random embedding $\mathbf{\Omega} \in \mathbb{R}^{m \times d}$ and construct a smaller *sketched* system.



A bold approach — "sketch-to-solve":

Find the vector $\mathbf{x}$ that solves the sketched system.

A safe approach — "sketch-to-precondition":

Build a *preconditioner* $\mathbf{M} \in \mathbb{R}^{n \times n}$ by factorizing $\mathbf{\Omega}^*\mathbf{A}$ so that $\mathbf{\Omega}^*\mathbf{A} = \mathbf{QM}$.

Iterate on the preconditioned linear system $(\mathbf{AM}^{-1})\,(\mathbf{Mx}) = \mathbf{b}$.

Provable reduction in computational complexity: From $O(mn^2)$ to $O(mn + n^3)$.

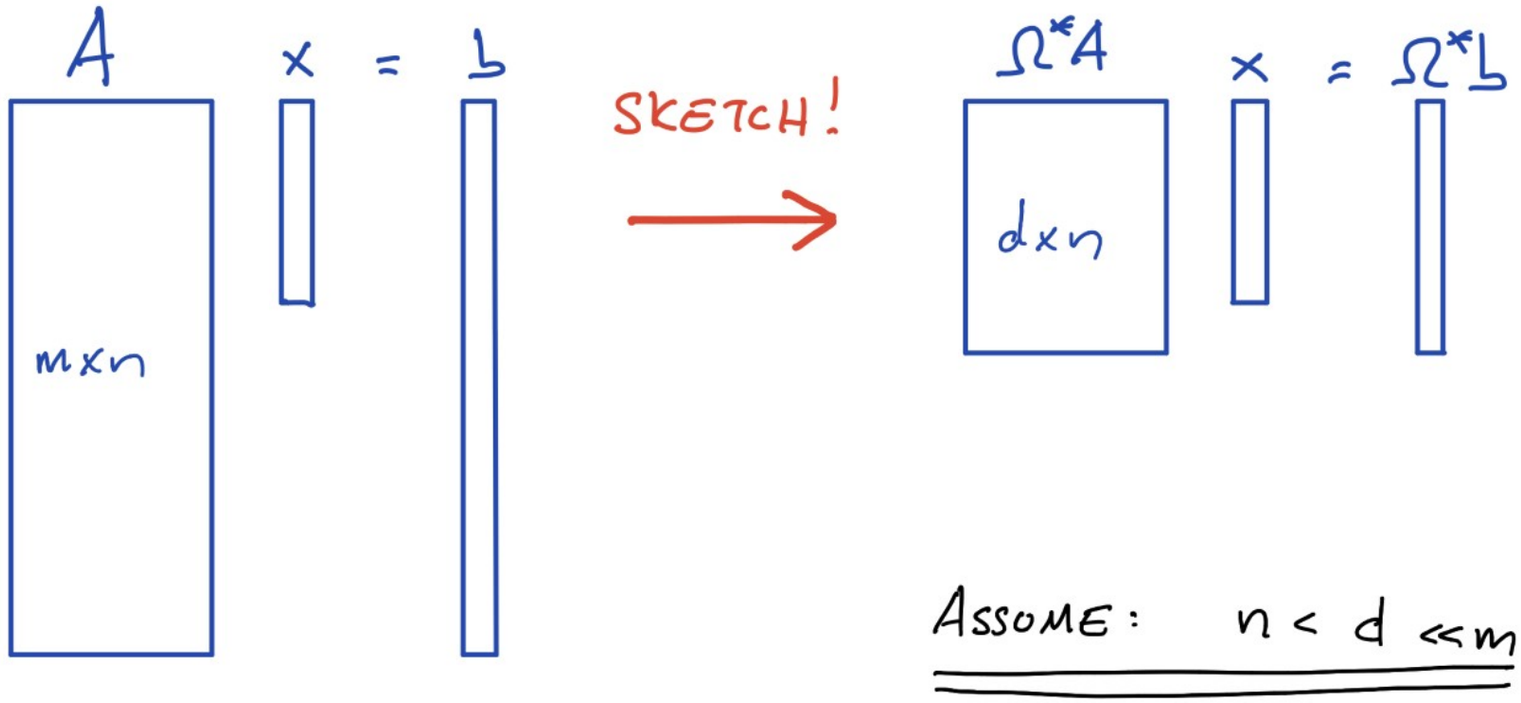
Ideal use of randomization. Guaranteed accuracy if you evaluate the residual.

Rokhlin/Tygert (2008), Avron/Maymounkov/Toledo (2010), many more

Random mixing for solving linear systems & least squares problems

Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ for $m \gg n$, and that you seek to solve $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$.

Draw a random embedding $\mathbf{\Omega} \in \mathbb{R}^{m \times d}$ and construct a smaller *sketched* system.

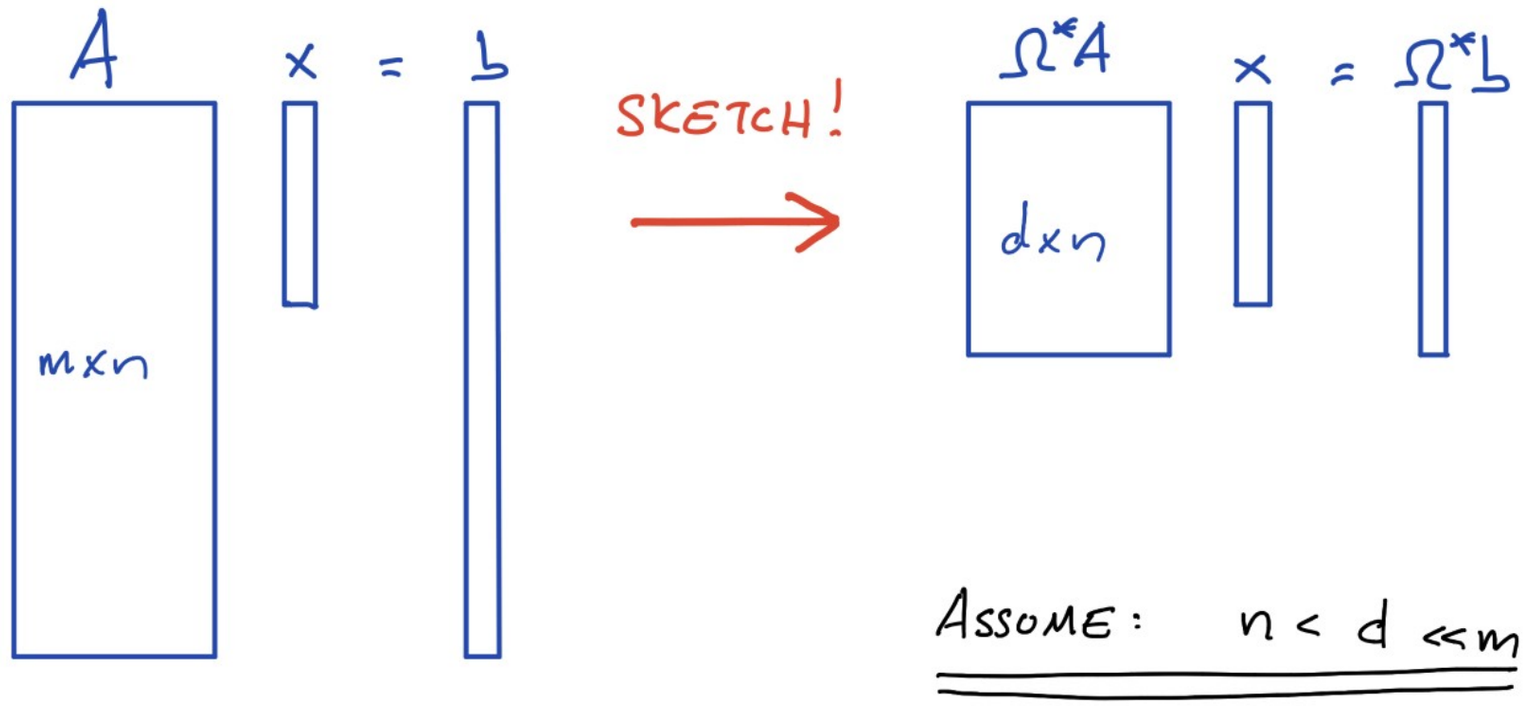


Question: Can the sketch be *down-sampling*? Simply pick d equations randomly?

Random mixing for solving linear systems & least squares problems

Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ for $m \gg n$, and that you seek to solve $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$.

Draw a random embedding $\mathbf{\Omega} \in \mathbb{R}^{m \times d}$ and construct a smaller *sketched* system.



Question: Can the sketch be *down-sampling*? Simply pick d equations randomly?

No, in general. Can fail catastrophically.

Yes, if you first randomly mix the equations.

Note: There are situations where randomized sampling is an essential tool – primarily when the whole matrix cannot be assembled. Examples include kernel ridge regression and certain gigantic linear systems arising in electronic structure calculations. "Down-sample and solve" is unavoidable in such cases.

It can also be very helpful when $\mathbf{A}$ has structure, as in tensor approximation.

Part 3: Global operators in scientific computing

A key challenge in scientific computing is that many of the operators we seek to model computationally are *global* in the sense that every point in the domain talks to every other point. In consequence, they need to be represented as *dense* matrices.

Examples:

- Solution operators of linear elliptic PDEs.
- Boundary-to-boundary operators (e.g. Dirichlet-to-Neumann).
- Time-evolution operators of parabolic PDEs.
- Scattering matrices for many wave propagation problems.
- Schur complements in sparse direct solvers.

Algorithms:

- Methods for *applying* global operators rapidly are well established in specialized cases (FFT, Fast Multipole Methods etc). Methods exist for more general operators, but this remains a topic of research (e.g. "$\mathcal{H}$-matrices").
- Techniques for *inverting, factorizing, exponentiating, ...* such operators exist, with progress ongoing — "$\mathcal{H}$-matrices", "Fast Direct Solvers", "inverse FMM" ...
- Today: Randomized methods for building data sparse representations.

Part 3: Global operators in scientific computing

A key challenge in scientific computing is that many of the operators we seek to model computationally are *global* in the sense that every point in the domain talks to every other point. In consequence, they need to be represented as *dense* matrices.

Examples:

- Solution operators of linear elliptic PDEs.
- Boundary-to-boundary operators (e.g. Dirichlet-to-Neumann).
- Time-evolution operators of parabolic PDEs.
- Scattering matrices for many wave propagation problems.
- Schur complements in sparse direct solvers.

Vision:

Create a framework where we can explicitly form data sparse representations of global linear operators, use them to model complex physical systems, and then *directly* solve the resulting system of equations.

Linear complexity direct solvers for many huge dense and sparse linear systems.

Example: Solution operators to linear elliptic boundary value problems

Consider a simple two-point boundary value problem

$$\text{(BVP)} \qquad \begin{cases} -u''(x) = g(x), & x \in \Omega = (0,1), \\ u(x) = 0, & x \in \Gamma = \partial\Omega = \{0,1\}. \end{cases}$$

We can write down the solution operator S explicitly as

$$\text{(SLN)} \qquad u(x) = [Sg](x) = \int_0^1 G(x,y)\, g(y)\, dy,$$

where the *Green's function* G takes the form

$$G(x,y) = \begin{cases} (1-x)y, & \text{when } x \geq y \quad \text{(on or below the diagonal)}, \\ x(1-y), & \text{when } x \leq y \quad \text{(on or above the diagonal)}. \end{cases}$$

Notes:

- The operator S is "benign" – compact, smoothing, ...
  (Cf. with the differential operator $d^2/(dx)^2$.)
- The operator S is *global* $\rightarrow$ *dense* matrix upon discretization.
- The kernel of S is (locally) *separable* $\rightarrow$ (locally) *low rank* matrices on discretization.
  Note the lack of regularity as you cross the diagonal, however.

Example: Solution operators to linear elliptic boundary value problems

Consider a simple elliptic PDE

$$\text{(PDE)} \qquad [-\Delta u](x) = g(x), \qquad x \in \Omega = \mathbb{R}^2,$$

combined with appropriate decay conditions at infinity. Again, we can write down the solution operator S explicitly:

$$\text{(SLN)} \qquad u(x) = [Sg](x) = \int_{\mathbb{R}^2} \phi(x-y)\, g(y)\, dy$$

where the *fundamental solution* ϕ takes the form $\phi(x-y) = \dfrac{1}{2\pi} \log |x-y|$.

Notes:

- The operator S is again more "benign" than the differential operator $-\Delta$.
- The operator S is *global* $\rightarrow$ *dense* matrix upon discretization.
- Is the kernel separable?

Example: Solution operators to linear elliptic boundary value problems

Consider a simple elliptic PDE

$$\text{(PDE)} \qquad [-\Delta u](x) = g(x), \qquad x \in \Omega = \mathbb{R}^2,$$

combined with appropriate decay conditions at infinity. Again, we can write down the solution operator S explicitly:

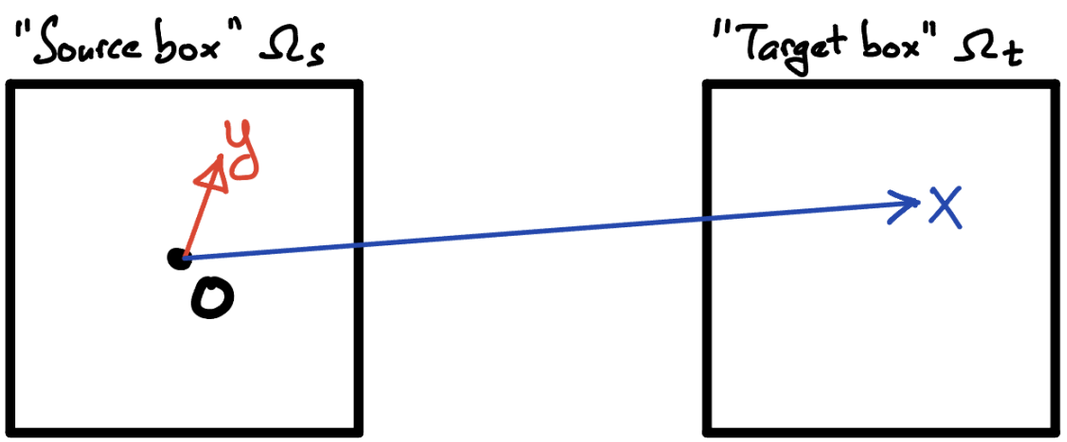
$$\text{(SLN)} \qquad u(x) = [Sg](x) = \int_{\mathbb{R}^2} \phi(x-y)\, g(y)\, dy$$

where the *fundamental solution* ϕ takes the form $\phi(x-y) = \dfrac{1}{2\pi} \log|x-y|$.

Approximate separability: Using complex notation, $x, y \in \mathbb{C}$, we have

$$\log(x-y) = \log(x) + \log(1-y/x) = \log(x) - \sum_{p=1}^{\infty} \frac{1}{p} \frac{y^p}{x^p}.$$

For given precision ε, truncate at $P \sim \log(1/\varepsilon) \to$ (local) separable expansion of rank P.



If $y \in \Omega_s$ and $x \in \Omega_t$, and $(\sqrt{2}/3)^P \leq \varepsilon$, then rank of interaction is $\leq P$ to precision ε.

Example: Solution operators to linear elliptic boundary value problems

Now consider a general linear boundary value problem of the form

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Examples of problems:

- The Laplace equation.
- The equations of linear elasticity.
- Stokes' equation.
- The Yukawa equation.
- Helmholtz' equation (at least at low and intermediate frequencies).
- Time-harmonic Maxwell (at least at low and intermediate frequencies).

Example: Solution operators to linear elliptic boundary value problems

Now consider a general linear boundary value problem of the form

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Linear solution operators: A general solution operator for (BVP) takes the form

$$\text{(SLN)} \qquad u(\boldsymbol{x}) = \int_\Omega G(\boldsymbol{x}, \boldsymbol{y})\, g(\boldsymbol{y})\, d\boldsymbol{y} + \int_\Gamma F(\boldsymbol{x}, \boldsymbol{y})\, f(\boldsymbol{y})\, dS(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Omega,$$

where G and F are two kernel functions that depend on A, B, and Ω.

Good: The operators in (SLN) are friendly and nice.

Bounded, smoothing, often fairly stable, etc.

Bad: The kernels G and F in (SLN) are generally *unknown.*

(Other than in trivial cases — constant coefficients and very simple domains.)

Bad: The operators in (SLN) are *global.*

Dense matrices upon discretization. $O(N^2)$ cost? $O(N^3)$ cost?

Example: Solution operators to linear elliptic boundary value problems

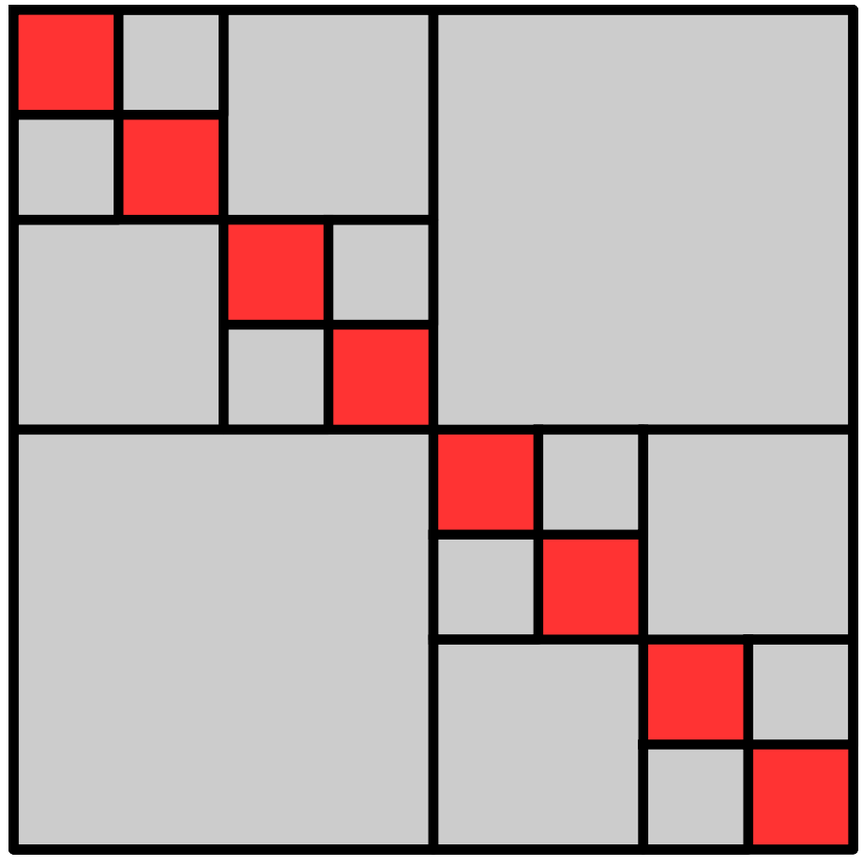
Now consider a general linear boundary value problem of the form

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Recurring idea: Upon discretization, (SLN) leads to a matrix with *off-diagonal blocks of low numerical rank.*

This property can be exploited to attain linear or close to linear complexity for operations such as matrix-vector multiply, matrix-matrix multiply, LU factorization, matrix inversion, forming Schur complements, operator exponentiation, etc.



All gray blocks have low rank.

Example: Solution operators to linear elliptic boundary value problems

Now consider a general linear boundary value problem of the form

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

In real life, tessellation patterns of rank structured matrices tend to be more complex ...

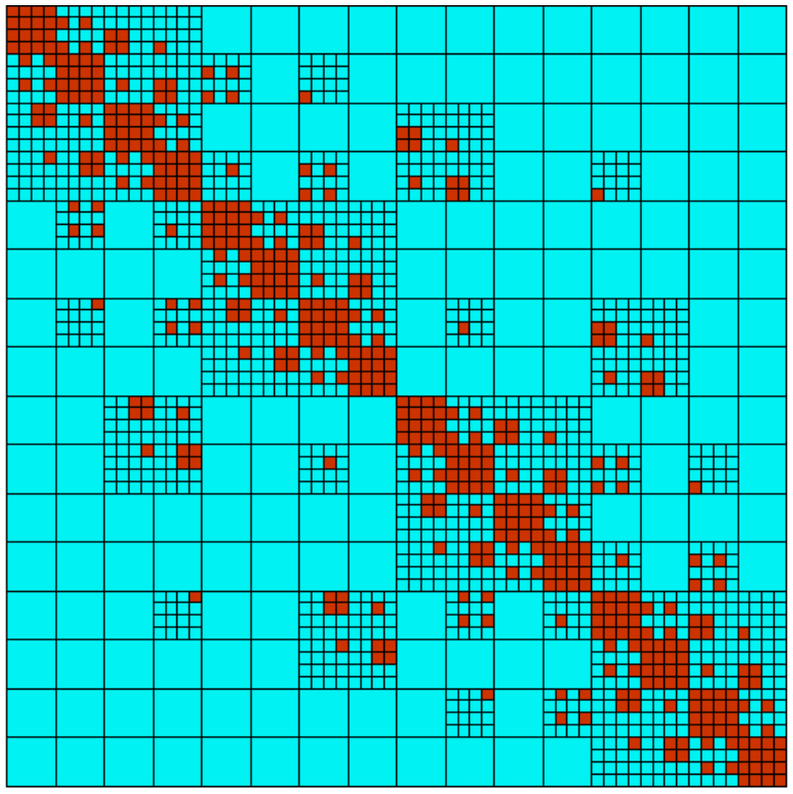


Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

Example: Solution operators to linear elliptic boundary value problems

Now consider a general linear boundary value problem of the form

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ Bu(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma, \end{cases}$$

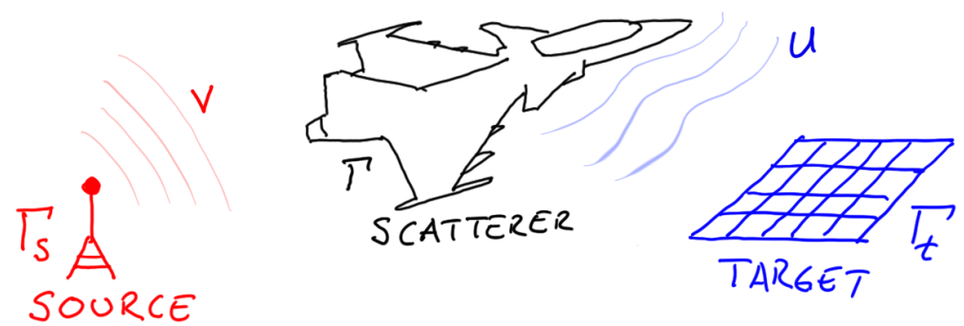
where Ω is a domain (2D or 3D) with boundary Γ, and where A is a linear elliptic differential operator; possibly with variable coefficients.

Strong connections to Calderón-Zygmund theory for singular integral operators.

References: Fast Multipole Method (Greengard, Rokhlin); Panel Clustering (Hackbusch); $\mathcal{H}$- and $\mathcal{H}^2$-matrices (Hackbusch et al); Hierarchically Block Separable matrices; Hierarchically Semi Separable matrices (Xia et al); HODLR matrices (Darve et al); BLR matrices (Buttari, Amestoy, Mary, . . .); . . .

Example: Boundary integral equation

Recall that many boundary value problems can advantageously be recast as *boundary integral equations.* Consider, e.g., (sound-soft) acoustic scattering from a finite body:



$$
(7) \qquad \begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 u(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^3 \backslash \overline{\Omega} \\ u(\boldsymbol{x}) = v(\boldsymbol{x}) & \boldsymbol{x} \in \partial\Omega \\ \lim\limits_{|\boldsymbol{x}|\to\infty} |\boldsymbol{x}| \left(\partial_{|\boldsymbol{x}|} u(\boldsymbol{x}) - i\kappa\, u(\boldsymbol{x}) \right) = 0. & \end{cases}
$$

The BVP (7) has an alternative mathematical formulation in the BIE

$$
(8) \qquad -\pi i \sigma(\boldsymbol{x}) + \int_{\partial\Omega} \left(\left(\partial_{\boldsymbol{n}(\boldsymbol{y})} + i\kappa \right) \frac{e^{i\kappa|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \right) \sigma(\boldsymbol{y})\, dS(\boldsymbol{y}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega.
$$

The integral equation (8) has several advantages over the PDE (7), including:

- The domain of computation $\partial\Omega$ is finite.
- The domain of computation $\partial\Omega$ is 2D, while $\mathbb{R}^3 \backslash \overline{\Omega}$ is 3D.
- Equation (8) is inherently well-conditioned (as a "2^{nd} kind Fredholm equation").

The integral operator (8) is global, and a matrix resulting from discretizing it is dense.

But both this matrix and its inverse are rank structured $\rightarrow O(N)$ solvers possible.

Example: Poincaré-Steklov operators (Dirichlet-to-Neumann, etc)

Consider a well posed linear boundary value problem

$$\text{(BVP)} \qquad \begin{cases} Au(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = f(\boldsymbol{x}), & \boldsymbol{x} \in \Gamma. \end{cases}$$

For $\boldsymbol{x} \in \Gamma$, let $h(\boldsymbol{x})$ denote the normal derivative of the solution u at $\boldsymbol{x}$. Then the map

$$T : f \mapsto h$$

is known as the *Dirichlet-to-Neumann (DtN)* map.

The DtN is a powerful tool in many areas of scientific computing:

- It provides a compressed representation that "hides" interior dynamics in a subdomain from the rest of the model. A mathematically "ideal" *reduced model*.
- Essential tool for understanding *domain decomposition* methods.
- The basis of many methods for *inverse problems* where you seek to reconstruct variable coefficients in A by observing input-output pairs.
- Ideal for coupling *multi-physics* problems – e.g. fluid-solid interactions.
- Etc etc.

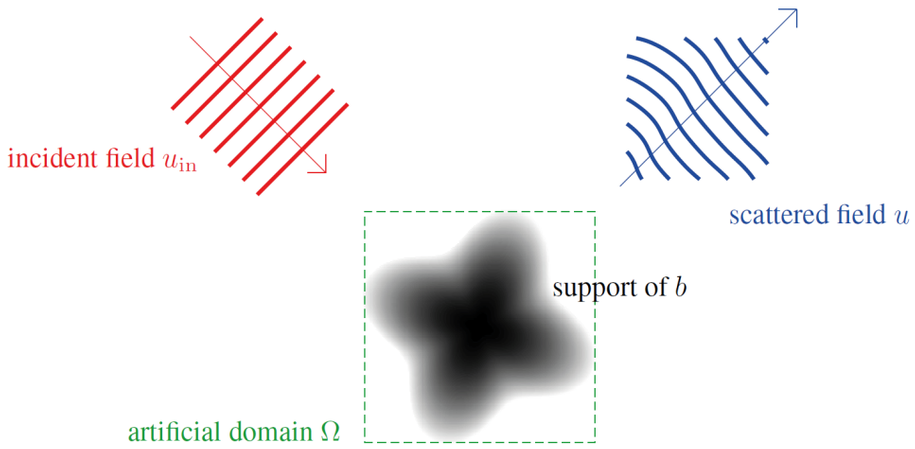
Example: Poincaré-Steklov operators (Dirichlet-to-Neumann, etc)

Consider the free space acoustic scattering problem

$$\begin{cases} -\Delta u(\boldsymbol{x}) - \kappa^2 \left(1 - b(\boldsymbol{x})\right) u(\boldsymbol{x}) = -\kappa^2 b(\boldsymbol{x})\, v(\boldsymbol{x}), & \boldsymbol{x} \in \mathbb{R}^2 \\ \lim\limits_{|\boldsymbol{x}|\to\infty} \sqrt{|\boldsymbol{x}|}\left(\partial_{|\boldsymbol{x}|} u(\boldsymbol{x}) - i\kappa\, u(\boldsymbol{x})\right) = 0, \end{cases}$$

where

- b is a smooth scattering potential with compact support, where
- v is a given "incoming potential" and where
- u is the sought "outgoing potential."



Introduce an artificial box Ω such that $\text{support}(b) \subseteq \Omega$.

On Ω:	On Ω^c:
• Variable coefficient PDE.	• Constant coefficient PDE.
• Discretize the PDE.	• Use BIE.
• Build DtN for $\partial\Omega$.	• Build DtN for $\partial\Omega^c$.

Glue the domains together using the DtNs.

(Actually, *impedance-to-impedance (ItI)* maps are better.)

[Gillman, Barnett, Martinsson, 2015]

Example: Schur complements in sparse linear solvers

When discretizing a linear elliptic PDE, one typically gets a linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the $N \times N$ matrix $\mathbf{A}$ is *sparse*. The most stable and robust linear solvers rely on techniques such as Gaussian elimination that directly build a factorization of the matrix

$$\mathbf{PAQ}^* = \mathbf{LU}$$

where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular, respectively. The matrices $\mathbf{P}$ and $\mathbf{Q}$ are permutation matrices that are chosen to balance numerical stability (avoiding large elements in $\mathbf{L}$ and $\mathbf{U}$) and loss of sparsity in the factors $\mathbf{L}$ and $\mathbf{U}$.

However, the asymptotic complexity of direct methods cannot be less than:

	Build stage	*Solve stage*
2D	$N^{3/2}$	$N \log N$
3D	N^2	$N^{4/3}$

The problem is that the *Schur complements* that arise in the factorization are dense, and get larger and larger as the computation proceeds.

Example: Schur complements in sparse linear solvers

When discretizing a linear elliptic PDE, one typically gets a linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the $N \times N$ matrix $\mathbf{A}$ is *sparse*. The most stable and robust linear solvers rely on techniques such as Gaussian elimination that directly build a factorization of the matrix

$$\mathbf{PAQ}^* = \mathbf{LU}$$

where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular, respectively. The matrices $\mathbf{P}$ and $\mathbf{Q}$ are permutation matrices that are chosen to balance numerical stability (avoiding large elements in $\mathbf{L}$ and $\mathbf{U}$) and loss of sparsity in the factors $\mathbf{L}$ and $\mathbf{U}$.

However, the asymptotic complexity of direct methods cannot be less than:

	Build stage		*Solve stage*	
2D	$N^{3/2}$	$\rightarrow N$	$N \log N$	$\rightarrow N$
3D	N^2	$\rightarrow N$	$N^{4/3}$	$\rightarrow N$

The problem is that the *Schur complements* that arise in the factorization are dense, and get larger and larger as the computation proceeds.

New: The Schur complements have rank structure. By exploiting this, linear or close to linear complexity is possible in all cases!

Interaction ranks: Why are they small?

We have claimed that a wide range of global operators that arise in scientific computing have rank structure. Specifically, the claim is that the numerical rank of interaction between two subdomains that are separated in space is low.

Why is this the case?

Interaction ranks: Why are they small?

We have claimed that a wide range of global operators that arise in scientific computing have rank structure. Specifically, the claim is that the numerical rank of interaction between two subdomains that are separated in space is low.

Why is this the case?

(One) Answer: It is a consequence of the *smoothing effect* of elliptic differential equations; it can be interpreted as a *loss of information.*

This effect has many well known physical consequences:

- Rapid convergence of *multipole expansions* when the region of sources is far away from the observation point.
- The *St Venant principle* in mechanics.
- The inaccuracy of imaging at sub-wavelength scales.
- The intractability of solving the heat equation backwards.

Caveat: High-frequency problems present difficulties — no loss of information for length-scales $> \lambda$. Extreme accuracy of optics, high-frequency imaging, *etc*.

Compression of a matrix discretizing a continuum operator

Question: How do you obtain a data sparse representation of a dense matrix discretizing a continuum operator?

Compression of a matrix discretizing a continuum operator

Question: How do you obtain a data sparse representation of a dense matrix discretizing a continuum operator?

- In some standard environments (convolution with a known fundamental solution, say), there exist analytic techniques that work very well. "Abramowitz & Stegun" or "proxy surface method". *Classical FMM environment*
- In some cases where the kernel is known explicitly, heuristic techniques such as "adaptive cross approximation" are often used. These are fast, but not 100% reliable. Ok for building pre-conditioners. *$\mathcal{H}$-matrix environment*

Compression of a matrix discretizing a continuum operator

Question: How do you obtain a data sparse representation of a dense matrix discretizing a continuum operator?

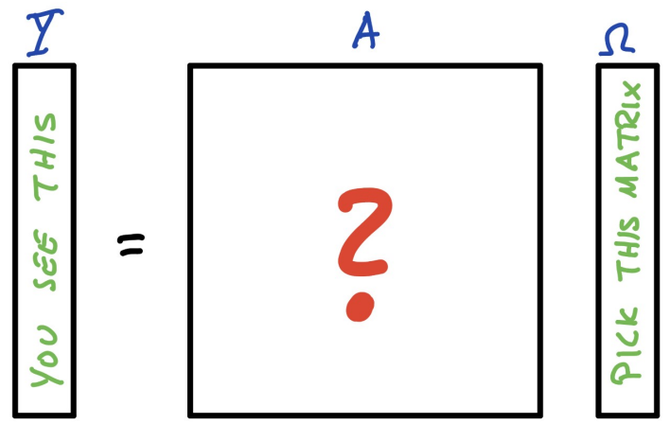
- In some standard environments (convolution with a known fundamental solution, say), there exist analytic techniques that work very well. "Abramowitz & Stegun" or "proxy surface method". *Classical FMM environment*
- In some cases where the kernel is known explicitly, heuristic techniques such as "adaptive cross approximation" are often used. These are fast, but not 100% reliable. Ok for building pre-conditioners. *$\mathcal{H}$-matrix environment*
- In more general cases, *randomized* algorithms are very competitive. These methods require that you have some means of applying the operator (e.g. a legacy PDE solver), so that you can observe input-output pairs.

Compression of a rank-structured matrix

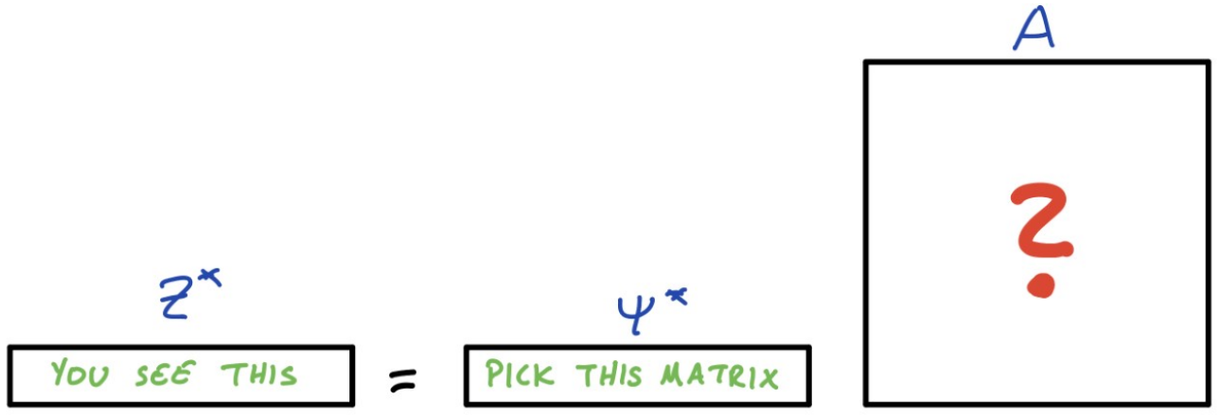
Environment: We are given a rank-structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

Sample the column space of the matrix:



If $\mathbf{A} \neq \mathbf{A}^*$, then sample the row space too:



In the present context, the application of $\mathbf{A}$ to $\mathbf{\Omega}$ typically involves running a legacy PDE solver, or applying some known fast method for the matrix-vector multiplication.

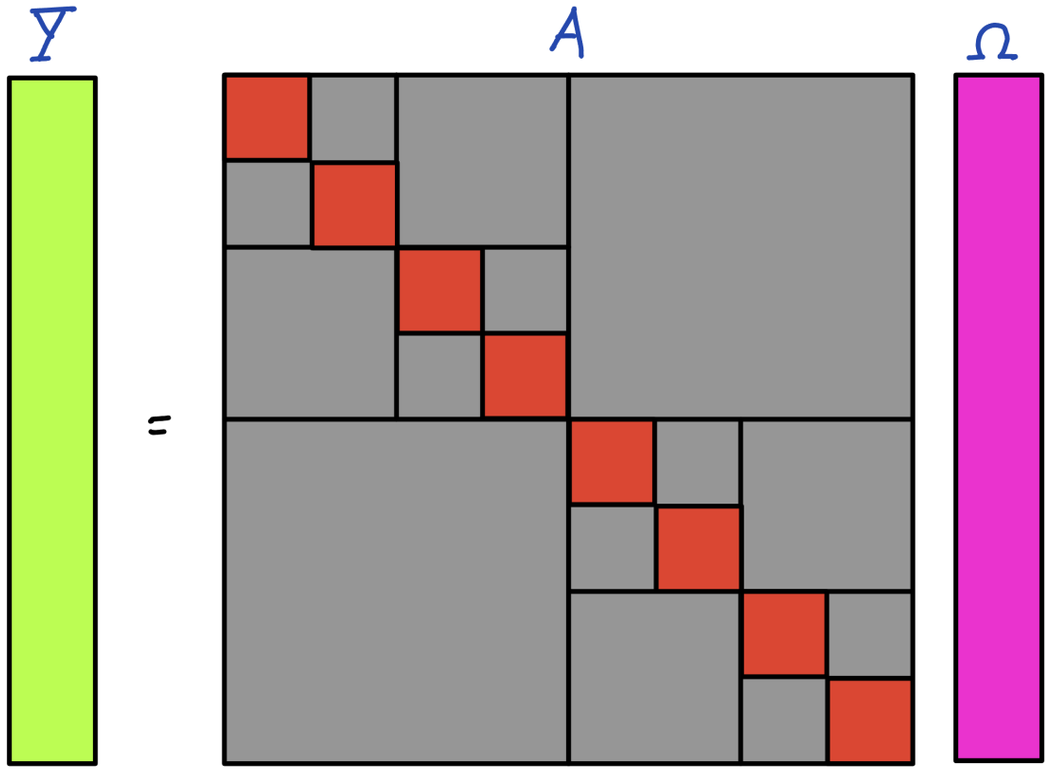
Compression of a rank-structured matrix

Environment: We are given a rank-structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

Objective: Construct thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ such that $\mathbf{A}$ can be completely reconstructed in $O(N)$ work from the set $\{\mathbf{Y}, \mathbf{\Omega}, \mathbf{Z}, \mathbf{\Psi}\}$ where $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

The Randomized SVD from Part 1 solves the problem when $\mathbf{A}$ has globally low rank.

The challenge in the rank structured case is that each part of the matrix $\mathbf{Y}$ contains a mix of components from many off-diagonal blocks, and from some dense blocks too.



How do you disentangle these components?

Compression of a rank-structured matrix: How do you do it?

1. **With entry evaluation:** If you can evaluate individual entries of the target matrix, then you can simply subtract off the dense blocks from the sample matrix.
   - First algorithms of this nature to be developed (2008).
   - Leads to optimal complexity, and high practical speed.
   - Restrictive assumption! Excludes many important applications.

2. **With zero blocks in test matrix:** Create test matrices with strategically placed zero blocks that hit the dense blocks of the target matrix.
   - First complete "black box" algorithms (2011).
   - Every level must be processed independently $\rightarrow O(N \log N)$ complexity.
   - Pre-factors tend to be large. Can be reduced through graph coloring algorithms that design bespoke zero patterns for any given geometry.

3. **With algebraic structure in the test matrix:** It turns out to be possible to form test matrices that allow you to retro-actively go back and create zeros blocks where you like *after* extracting a universal sample.
   - High practical speed, and first method that is $O(N)$ and completely black box (2022).
   - Original method works best for 2D problems, or problems on surfaces in 3D.
   - Current work is aimed at finding different algebraic structures that lead to high efficiency for fully general 3D geometries.
   - It turns out to be possible to *invert* the matrix as you go!

Compression of a rank-structured matrix: How do you do it?

1. **With entry evaluation:** If you can evaluate individual entries of the target matrix, then you can simply subtract off the dense blocks from the sample matrix.

   *[P.G. Martinsson, SIMAX, **32**(4), 2011. Later improvements by Jianlin Xia, Xiaoye Sherry Li, . . .*]

2. **With zero blocks in test matrix:** Create test matrices with strategically placed zero blocks that hit the dense blocks of the target matrix.

   *[L. Lin, J. Lu, L. Ying, JCP, **230**(10), pp. 4071–4087, 2011; P.G. Martinsson, SISC, **38**(4), pp. A1959-A1986, 2016;*

   *J. Levitt & P.G. Martinsson, JCAM **451**(1), 2024.]*

3. **With algebraic structure in the test matrix:** It turns out to be possible to form test matrices that allow you to retro-actively go back and create zeros blocks where you like *after* extracting a universal sample.

   *[J. Levitt & P.G. Martinsson, SISC, **46**(3), 2024. K. Pearce, A. Yesypenko, J. Levitt, P.G. Martinsson, to appear in SIMAX*

   (arXiv:2501.05528). A. Yesypenko & P.G. Martinsson, to appear in J. Sci. Comp. (arxiv:2311.01451).]

4. **Plenty of related work recently!**

   Exploiting the continuum structure of the underlying operator: A. Townsend & N. Boullé (2022).

   Complexity analysis – what is the minimal number of matvecs required? Recent work by A. Townsend, D. Halikias, C. Musco & C. Musco, D. Persson, N. Boullé.

   Randomized compression of butterfly matrices: Path towards medium/high frequency problems. *[Y. Liu, X. Xing, H. Guo, E. Michielssen, P. Ghysels, X.S. Li. 2021], [Y. Li, H. Yang, E. Martin, K. Ho, and L. Ying, 2015].*

Key points

- Randomized algorithms with high accuracy and reliability. “Las Vegas” style.
- Practical win: Less communication $\rightarrow$ far higher speed.
- Theory wins: Streaming algorithms, dramatic improvement in asymptotic costs.
  Supported by random matrix theory and high dimensional probability theory.
- Extension to global operators: Continuum problems, hierarchical matrices.
  Solution operators, evolution operators, DtN maps, $O(N)$ sparse direct solvers, . . .

Randomness is not a source of error here; it’s the engine of algorithmic efficiency.

Surveys:

- P.G. Martinsson and M. O'Neil, "Fast Direct Solvers for Elliptic PDEs".
  Review of direct algorithms for global operators in scientific computing. (Arxiv report 2511.07773)
- K. Pearce and P.G. Martinsson, "Randomized Alg. for Low-Rank Matrix and Tensor Decompositions".
  Review of randomized methods for compression of tensors. (Arxiv report 2512.05286)
- P.G. Martinsson and J. Tropp, "Randomized Numerical Linear Algebra: Foundations & Algorithms".
  Acta Numerica, 2020. (Arxiv report 2002.01387)
  Long survey summarizing major findings in the field in the past decade.
- P.G. Martinsson, "Randomized methods for matrix computations." The Mathematics of Data, IAS/Park City Mathematics Series, 25(4), pp. 187 - 231, 2018.
  Book chapter that is written to be accessible to a broad audience. Focused on practical aspects.
- N. Halko, P.G. Martinsson, J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Review*, 53(2), 2011, pp. 217-288.
  Survey that describes the randomized SVD and its variations.

Tutorials, summer schools, etc:

- 2020: 3 lecture mini course on randomized linear algebra, KTH, Stockholm. Videos available.
- 2016: Park City Math Institute (IAS): *The Mathematics of Data.*
- 2014: CBMS summer school at Dartmouth College. 10 lectures on YouTube.
- 2009: NIPS tutorial lecture, Vancouver, 2009. Online video available.

DOE report on randomized algorithms: `https://arxiv.org/abs/2104.11079` (2021)

Slides: https://users.oden.utexas.edu/∼pgm/