Randomization in computational linear algebra

Gunnar Martinsson

Dept. of Mathematics & Oden Institute for Computational Sciences and Engineering

University of Texas at Austin

Collaborators: Robert van de Geijn, Abinand Gopal, Francisco Igual, Yuji Nakatsukasa, Gregorio Quintana-Ortí, Vladimir Rokhlin, Joel Tropp, Mark Tygert.

Current and recent students/postdocs: Yunhui Cai, Chao Chen, Ke Chen, Simon Dirckx, Yijun Dong, Nathan Heavner, Joseph Kump, James Levitt, Kate Pearce, Heather Wilber, Anna Yesypenko.

Research support by:

Outline

- **Methods based on randomized projections** *Main topic.*
  - Low rank approximation: The randomized SVD.
  - Streaming and single-pass methods.
  - A posteriori error estimation.
  - Structured random maps (“fast Johnson-Lindenstrauss transforms”).
  - Linear solvers: Sketch-to-solve vs. Sketch-to-precondition.
  - Las Vegas style methods: Extremely robust and reliable.

- **Methods based on randomized sampling** *Time permitting . . .*
  - Monte Carlo style methods → less reliable, less accurate, less robust.
  - Enable the solution of stupendously large problems that would otherwise be intractable.

- **Approximation of rank-structured matrices** *Very brief!*
  - Approximation of global operators of mathematical physics (solution operators, DtNs, . . .).
  - Tools for matrices that are not of global low rank, but have structure that can be exploited.
  - Conceptually related to Fast Multipole Methods, Fast Direct Solvers, Calderón-Zygmund theory . . .

Low rank approximation — problem formulation:

Let $\mathbf{A}$ be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll \min(m, n)$.

We seek to compute approximate factors $\mathbf{E}$ and $\mathbf{F}$ such that

$$\begin{array}{cccc} \mathbf{A} & \approx & \mathbf{E} & \mathbf{F}^*. \\ m \times n & & m \times k & k \times n \end{array}$$

Low rank approximation — problem formulation:

Let $\mathbf{A}$ be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll \min(m, n)$.

We seek to compute approximate factors $\mathbf{E}$ and $\mathbf{F}$ such that

$$\underset{m \times n}{\mathbf{A}} \quad \approx \quad \underset{m \times k}{\mathbf{E}} \;\; \underset{k \times n}{\mathbf{F}^*}.$$

Why?

- Fitting a hyperplane to a given set of points. Or fitting a multivariate normal distribution to measurements (“principal component analysis”).
- Model reduction in scientific computing.
- Spectral algorithms in data analysis.
- “Fast” algorithms of various types: Fast Multipole Methods, generalizations of the Fast Fourier Transform, fast direct solvers, etc.
- Many, many, many more.

Note: We seek only to control the residual error $\|\mathbf{A} - \mathbf{E}\mathbf{F}^*\|$.

Low rank approximation — problem formulation:

Let $\mathbf{A}$ be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll \min(m,n)$.

We seek to compute approximate factors $\mathbf{E}$ and $\mathbf{F}$ such that

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{E} & \mathbf{F}^*. \\ m \times n & & m \times k & k \times n \end{array}$$

Why?

- Fitting a hyperplane to a given set of points. Or fitting a multivariate normal distribution to measurements (“principal component analysis”).
- Model reduction in scientific computing.
- Spectral algorithms in data analysis.
- “Fast” algorithms of various types: Fast Multipole Methods, generalizations of the Fast Fourier Transform, fast direct solvers, etc.
- Many, many, many more.

Note: We seek only to control the residual error $\|\mathbf{A} - \mathbf{E}\mathbf{F}^*\|$.

Precise problem formulation: Given $\mathbf{A}$ (and possibly k), find an approximate orthonormal basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for the range of $\mathbf{A}$, so that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$, where $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_k]$.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{*}\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^k$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \big[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\big]$. The rank k may or may not be known.

If you have $\mathbf{Q}$, many "low-rank approximation" tasks become easy.

Example: Suppose $\mathbf{A}$ is symmetric, and you seek an approximate eigenvalue decomposition (EVD). Form the matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}\mathbf{Q}$, and compute the EVD of $\mathbf{B}$:

$$\mathbf{B} = \mathbf{V}\mathbf{D}\mathbf{V}^*.$$

Observe that $\mathbf{B}$ is of size $k \times k$. *Small!*

Then

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}\mathbf{Q}\mathbf{Q}^* = \mathbf{Q}\mathbf{B}\mathbf{Q}^* = \mathbf{Q}\mathbf{V}\mathbf{D}\mathbf{V}^*\mathbf{Q}^* = \{\text{Set } \mathbf{U} := \mathbf{Q}\mathbf{V}\} = \mathbf{U}\mathbf{D}\mathbf{U}^*$$

is an approximate EVD of $\mathbf{A}$.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \big[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\big]$. The rank k may or may not be known.

Excellent algorithms for computing $\mathbf{Q}$ already exist:

- **Compute the full SVD and truncate**
  - Eckart-Young theorem tells us this is optimal.
  - Algorithms are stable, accurate, and reliable. Routinely yield double precision accuracy.
  - Algorithms must be iterative, but converge *extremely* fast in practice.
  - Cost scales poorly with matrix size, $O(n^3)$ for $n \times n$ matrix. Hard to parallelize.
- **Krylov methods** For instance, $\mathrm{ran}(\mathbf{Q}) = \mathrm{ran}\big(\big[\mathbf{b},\, \mathbf{A}\mathbf{b},\, \mathbf{A}^2\mathbf{b},\, \dots,\, \mathbf{A}^{k-1}\mathbf{b}\big]\big)$.
  - Particularly efficient when fast matrix-vector multiplication is available. (E.g. $\mathbf{A}$ is sparse.)
  - If the metric is number of matvecs, then Krylov methods are often “optimal”.
  - Well understood – precise theory is available in many (but far from all!) cases.
- **Gram-Schmidt (with “column pivoting”)**
  - Very simple!
  - When column pivoting is used, it is robust and typically works well.
  - Can be stopped after k steps if required tolerance is met.
  - Communication intensive, so somewhat slow in practice. Does not preserve sparsity.
  - Output is quite far from optimal when singular values decay slowly.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \big[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\big]$. The rank k may or may not be known.

Let us consider Gram-Schmidt in some detail, applied to $\mathbf{A} = \big[\mathbf{a}_1 \ \mathbf{a}_2 \cdots \mathbf{a}_n\big]$.

In the “generic” case, Gram-Schmidt proceeds as follows:

$$\begin{array}{lll}
\mathbf{a}_1' = \mathbf{a}_1 & \mathbf{q}_1 = \dfrac{1}{\|\mathbf{a}_1'\|}\mathbf{a}_1' & \mathbf{P}_1 = \mathbf{q}_1\mathbf{q}_1^* \\
\mathbf{a}_2' = (\mathbf{I} - \mathbf{P}_1)\mathbf{a}_2 & \mathbf{q}_2 = \dfrac{1}{\|\mathbf{a}_2'\|}\mathbf{a}_2' & \mathbf{P}_2 = \mathbf{P}_1 + \mathbf{q}_2\mathbf{q}_2^* \\
\mathbf{a}_3' = (\mathbf{I} - \mathbf{P}_2)\mathbf{a}_3 & \mathbf{q}_3 = \dfrac{1}{\|\mathbf{a}_3'\|}\mathbf{a}_3' & \mathbf{P}_3 = \mathbf{P}_2 + \mathbf{q}_3\mathbf{q}_3^* \\
\vdots & \vdots & \vdots
\end{array}$$

The algorithm can fail outright, since any vector $\mathbf{a}_j'$ may be zero.

It can also fail numerically, if $\mathbf{a}_j'$ is small. (Fixable, by swapping to *unitary* transforms.)

Classical fix: Column pivoting – apply projections to all columns, and pick the largest remaining one. Greatly increases cost! And ruins sparsity.

Randomization fix: If you randomly mix the columns first, you can forgo pivoting!

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^k$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \big[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\big]$. The rank k may or may not be known.

Gram-Schmidt with randomized mixing:

Draw an $n \times n$ matrix $\mathbf{\Omega}$ from a Gaussian distribution, and form

$$\mathbf{Y} = \big[\mathbf{y}_1\ \mathbf{y}_2 \cdots \mathbf{y}_n\big] = \mathbf{A}\mathbf{\Omega}.$$

Then perform Gram-Schmidt on the vectors $\{\mathbf{y}_j\}_{j=1}^k$ to form the ON set $\{\mathbf{q}_j\}_{j=1}^k$.

Claim: The matrix $\mathbf{Q} = \big[\mathbf{q}_1\ \mathbf{q}_2 \cdots \mathbf{q}_k\big]$ is often a good answer to our “task”.

A basic analysis of the scheme is easy. Let $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^*$ be the exact SVD of $\mathbf{A}$. Then

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

Observe that $\mathbf{\Psi}$ *also* has a Gaussian distribution. We find that

$$\mathbf{y}_i = \mathbf{Y}(:,i) = \mathbf{U}\mathbf{D}\mathbf{\Psi}(:,i) = \sum_{j=1}^{n} \psi_{j,i}\,\sigma_j\,\mathbf{u}_j.$$

Cost of scheme: $O(mk^2)$ flops + the cost of forming $\mathbf{A}\mathbf{\Omega}$. (So $O(mnk)$ in general.)

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^k$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Algorithm: (Assuming k is given in advance.)

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

Recall that

$$\mathbf{y}_i = \sum_{j=1}^{n} (\sigma_j \psi_{ji})\, \mathbf{u}_j, \qquad \text{where } \psi_{ji} \sim \mathcal{N}(0,1).$$

Quick facts:

- When $\mathbf{A}$ has *exact* rank k, the algorithm succeeds (i.e. $\mathbf{A} = \mathbf{Q}\mathbf{Q}^*\mathbf{A}$) with probability 1.
- When the singular values *decay rapidly*, we expect close to optimal results.
- When the singular values *decay slowly*, the basis will likely not be good.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Algorithm: (Assuming k is given in advance.)

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `Y = A * Omega`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

Recall that

$$\mathbf{y}_i = \sum_{j=1}^{n} \left(\sigma_j \psi_{ji}\right) \mathbf{u}_j, \qquad \text{where } \psi_{ji} \sim \mathcal{N}(0,1).$$

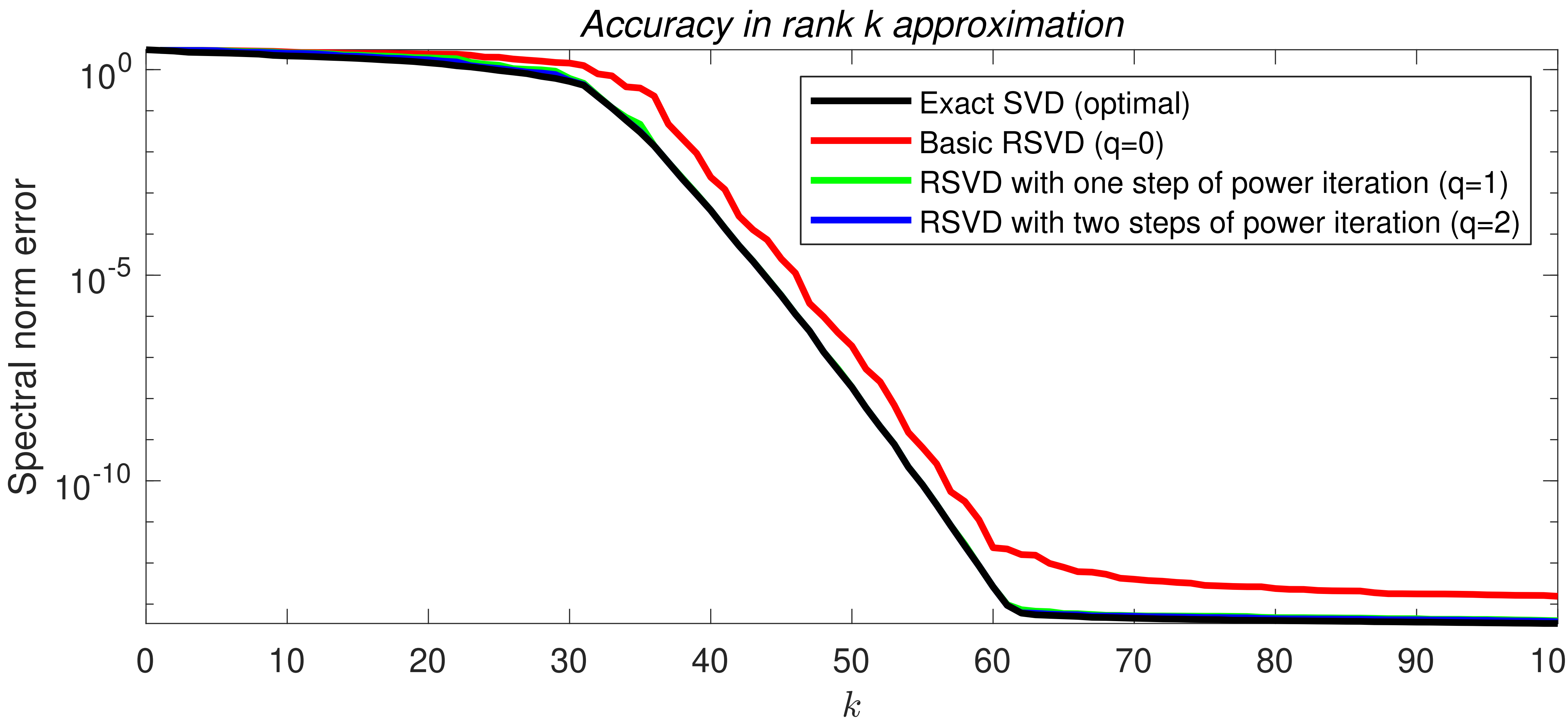
Quick facts:

- When $\mathbf{A}$ has *exact* rank k, the algorithm succeeds (i.e. $\mathbf{A} = \mathbf{Q}\mathbf{Q}^*\mathbf{A}$) with probability 1.
- When the singular values *decay rapidly*, we expect close to optimal results.
- When the singular values *decay slowly*, the basis will likely not be good.

  This can be fixed via powering, to boost the separation between large/small svds.

  For instance, replace $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ *by* $\mathbf{Y} = \mathbf{A}\mathbf{A}^*\mathbf{A}\mathbf{\Omega}$. *Then* $\mathbf{y}_i = \sum_{j=1}^{n}\left(\sigma_j^3 \psi_{ji}\right) \mathbf{u}_j$.

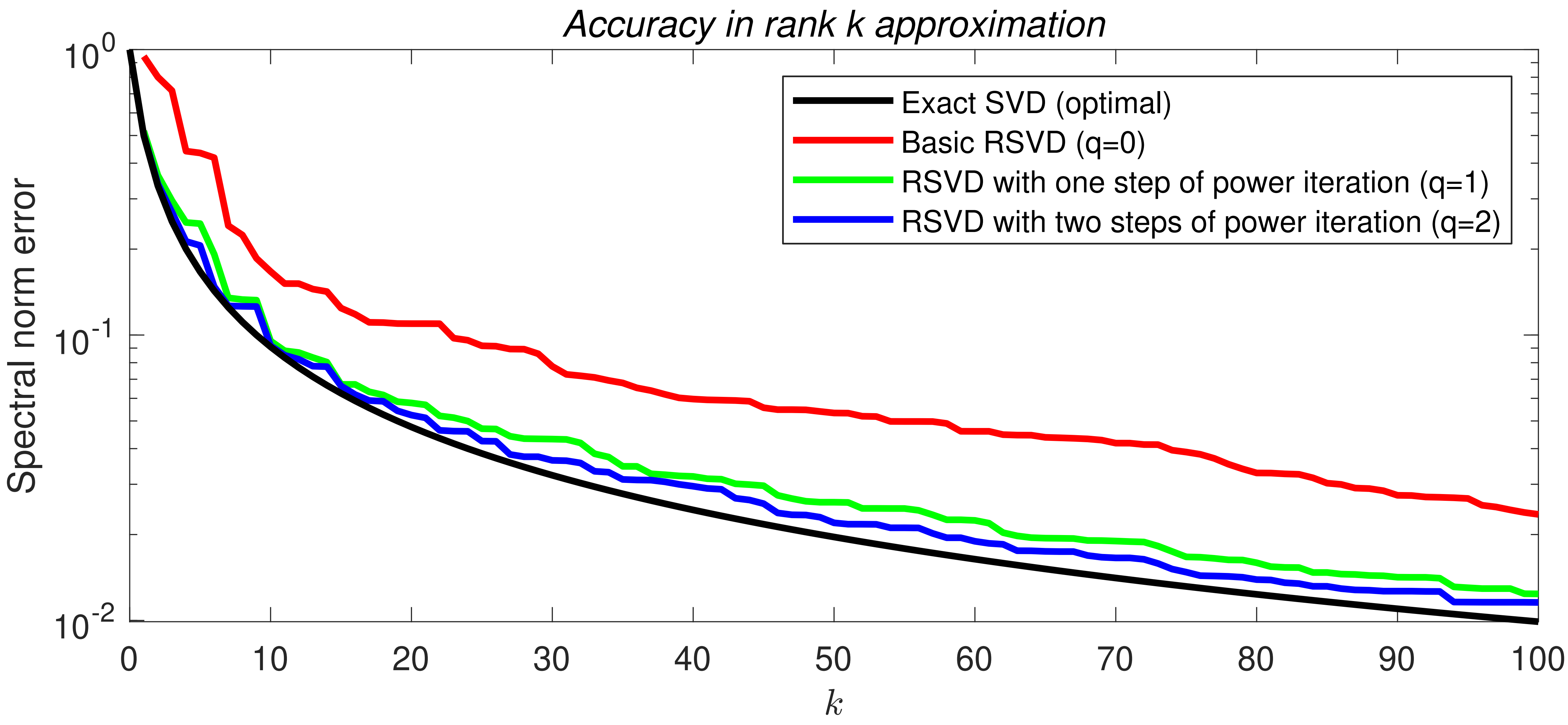
Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.



The plot shows the errors $e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|$, 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. (For clarity, no oversampling is done.)

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

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

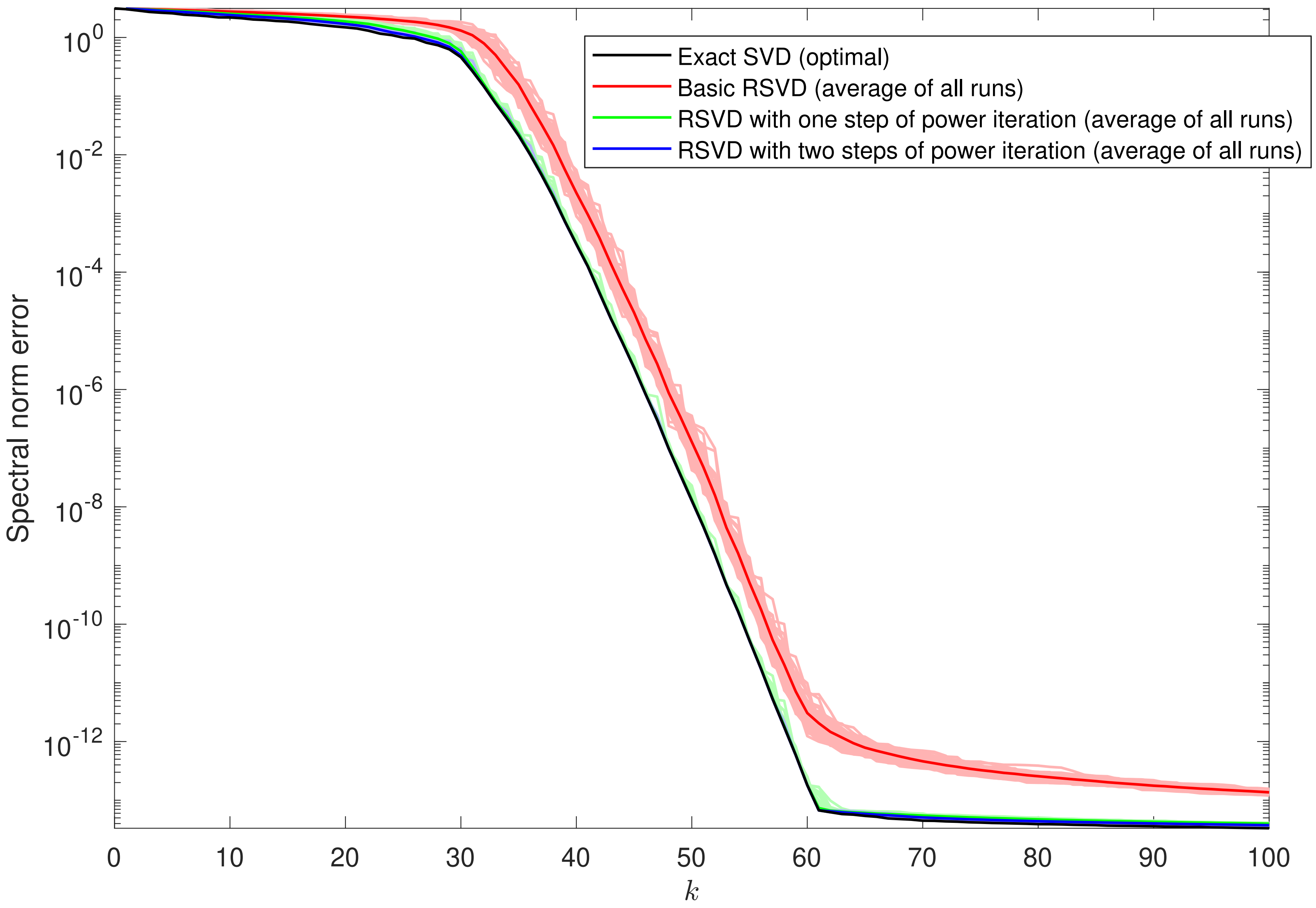


The plot shows the errors $e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|$, where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of $\mathbf{Y} = \left(\mathbf{A}\mathbf{A}^*\right)^q\mathbf{A}\boldsymbol{\Omega}$, and where $\boldsymbol{\Omega}$ is a Gaussian random matrix. (For clarity, no oversampling is done.)

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

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

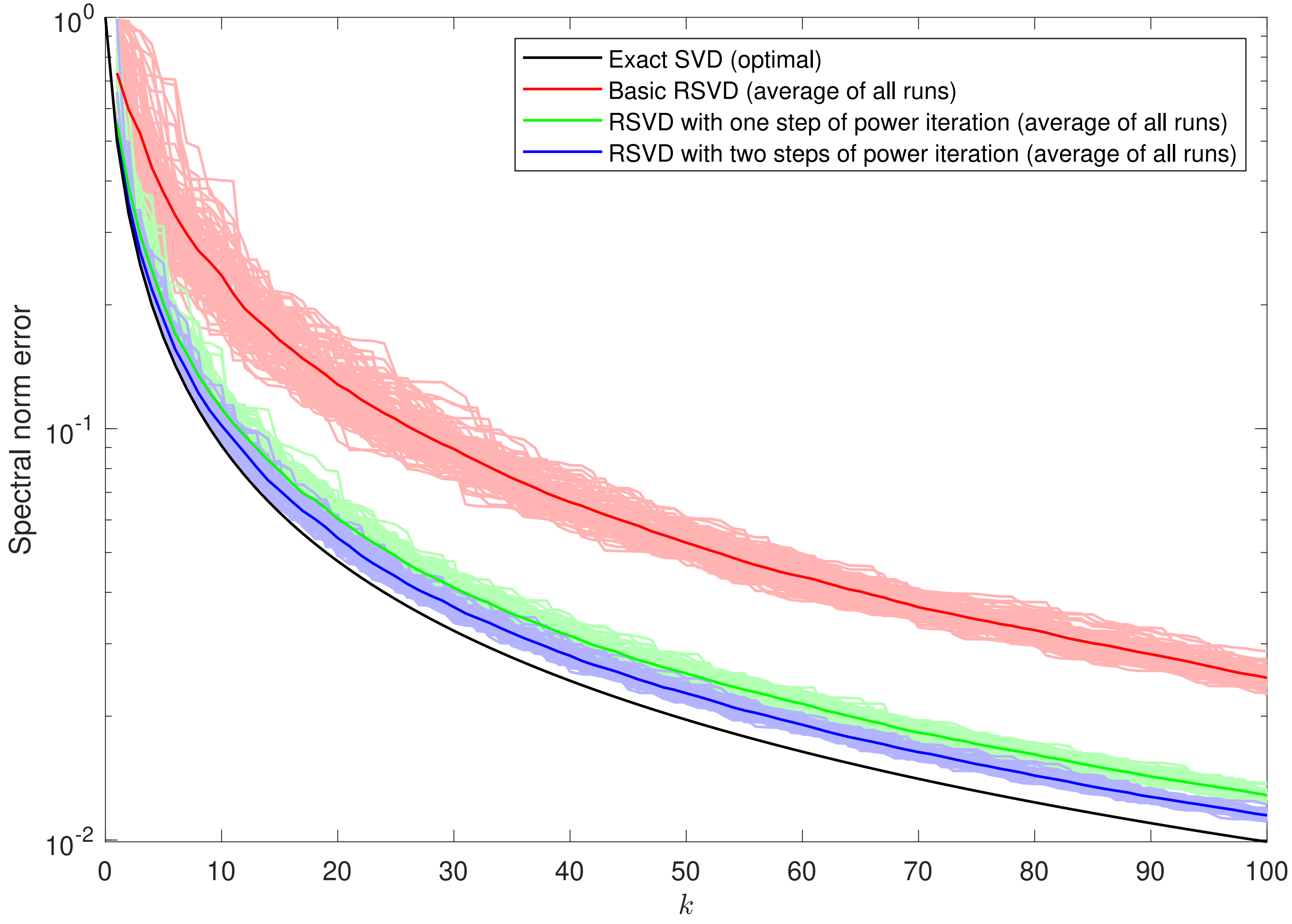
The same plot as before, but now showing 100 instantiations.



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

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{*}\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

The same plot as before, but now showing 100 instantiations.



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

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{*}\mathbf{A}\|$ small where $\mathbf{Q} = \left[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\right]$. The rank k may or may not be known.

The probabilistic behavior of the error has been studied extensively, and reasonably tight theoretical bounds have been established.

A small sample of references addressing this and related problems:

- Frieze, Kannan & Vempala, 2004.
- Martinsson, Rokhlin & Tygert,YALEU/DCS/RR-1361, 2006.
- Liberty, Woolfe, Martinsson, Rokhlin & Tygert, PNAS, 2008.
- Halko, Martinsson & Tropp, *SIAM Review*, 2011.
- Witten & Candès, *Algorithmica*, 2015.
- Gu, *SISC*, 2015. Analysis of randomized subspace iteration.
- Musco & Musco, *NIPS*, 2015. Analysis of block Krylov methods.
- Saibaba, *SIMAX*, 2019. Accuracy of singular vectors.
- Martinsson & Tropp, *Acta Numerica*, 2020.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = [\mathbf{q}_1 \ \cdots \ \mathbf{q}_k]$. The rank k may or may not be known.

To illustrate, suppose that we seek to match the optimal error in a rank-k approximation. We apply the basic randomized procedure involving $\ell \geq k$ samples.

1. Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$. `Omega = randn(n,l)`
2. Form the $m \times \ell$ sample matrix $\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}$. `Y = A * Omega`
3. Form an $m \times \ell$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

The resulting error then satisfies

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\boldsymbol{\Psi}_{(n-k),\ell}\boldsymbol{\Psi}_{k,\ell}^{\dagger}\|^2$$

where "$\boldsymbol{\Psi}_{i,j}$" denotes an $i \times j$ matrix drawn from a Gaussian distribution, and where $\mathbf{D}_2 = \text{diag}(\sigma_{k+1}, \sigma_{k+2}, \dots, \sigma_{\min(m,n)})$ holds the "tail" singular values of $\mathbf{A}$.

Note: The term $\|\mathbf{D}_2\|^2$ is the minimal error, according to Eckart-Young.

The suboptimality term depends on $\sigma_{\min}(\boldsymbol{\Psi}_{k,\ell})$, where $\boldsymbol{\Psi}_{k,\ell}$ is a $k \times \ell$ Gaussian matrix.

- With no oversampling, $k = \ell$, we get terrible results!
- With lots of oversampling, $k \ll \ell$, things look good. ("Marchenko–Pastur" case)

What about the case where ℓ is only slightly larger than k?

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \big[\mathbf{q}_1 \ \cdots \ \mathbf{q}_k\big]$. The rank k may or may not be known.

To illustrate, suppose that we seek to match the optimal error in a rank-k approximation. We apply the basic randomized procedure involving $\ell \geq k$ samples.

1. Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$. `Omega = randn(n,l)`
2. Form the $m \times \ell$ sample matrix $\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}$. `Y = A * Omega`
3. Form an $m \times \ell$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

The resulting error then satisfies

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\boldsymbol{\Psi}_{(n-k),\ell}\boldsymbol{\Psi}_{k,\ell}^{\dagger}\|^2$$

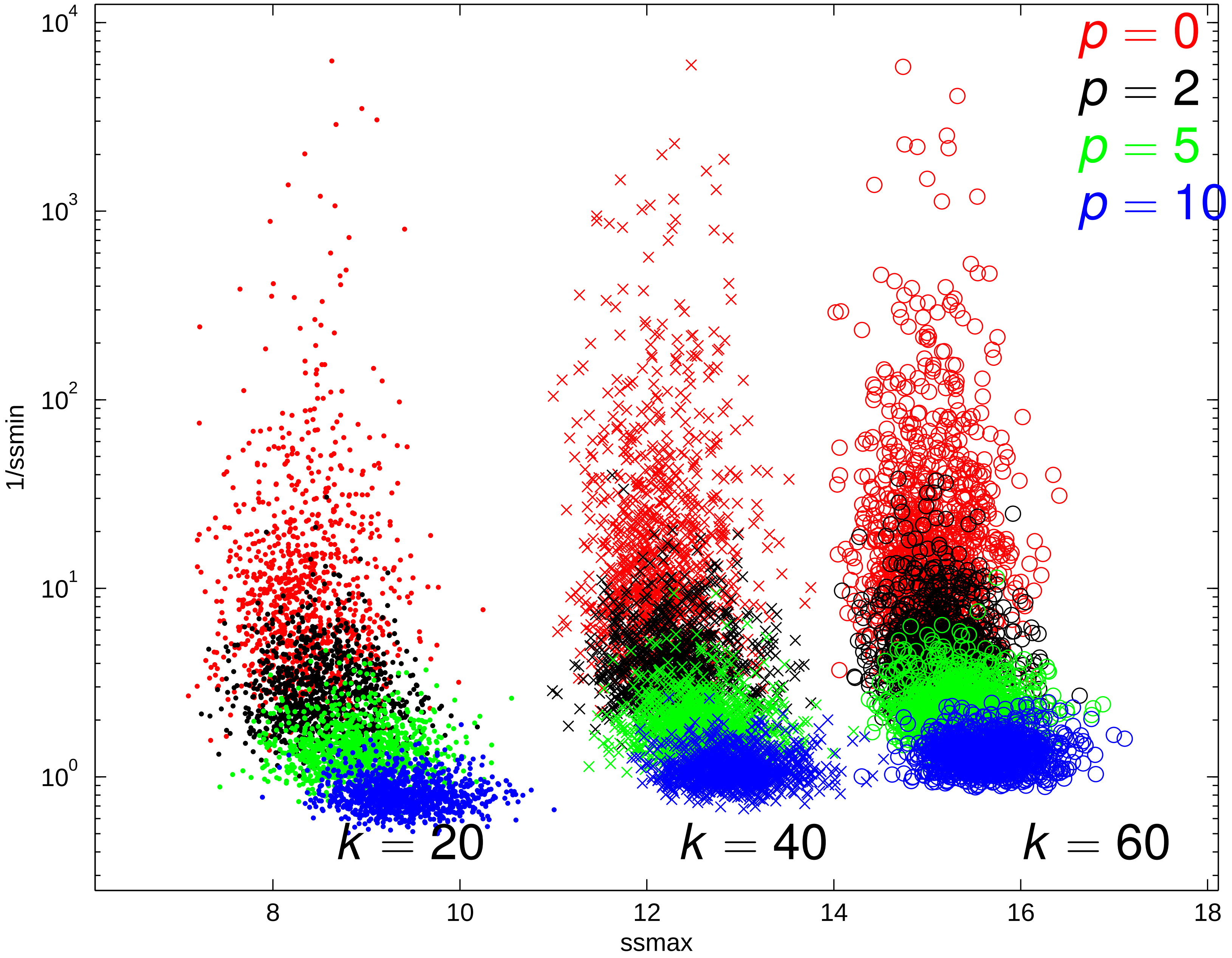
where "$\boldsymbol{\Psi}_{i,j}$" denotes an $i \times j$ matrix drawn from a Gaussian distribution, and where $\mathbf{D}_2 = \text{diag}(\sigma_{k+1},\, \sigma_{k+2},\, \dots,\, \sigma_{\min(m,n)})$ holds the "tail" singular values of $\mathbf{A}$.

Note: The term $\|\mathbf{D}_2\|^2$ is the minimal error, according to Eckart-Young.

Proposition: (Chen & Dongarra 2005) Let $\boldsymbol{\Psi}$ be a Gaussian matrix of size $k \times k+p$ where $p \geq 2$. Then $\big(\mathbb{E}\big[\|\boldsymbol{\Psi}^{\dagger}\|_{\mathrm{F}}^{2}\big]\big)^{1/2} \leq \sqrt{\frac{k}{p-1}}$ and $\mathbb{E}\big[\|\boldsymbol{\Psi}^{\dagger}\|\big] \leq \frac{e\sqrt{k+p}}{p}$.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = [\mathbf{q}_1 \ \cdots \ \mathbf{q}_k]$. The rank k may or may not be known.

Scatter plot showing distribution of $1/\sigma_{\rm min}$ for $k \times (k+p)$ Gaussian matrices.



$1/\sigma_{\rm min}$ is plotted against $\sigma_{\rm max}$.

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Using the bound $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\mathbf{\Psi}_{(n-k),\ell}\mathbf{\Psi}_{k,\ell}^{\dagger}\|^2$, along with the bounds on $\sigma_{\min}(\Psi_{k,\ell})$, we can bound the expectation of the error:

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 set $\mathbf{Q} = \texttt{orth}(\mathbf{A\Omega})$. Then the average error satisfies

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

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

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

(It turns out to decay super-exponentially fast as p increases!)

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^{*}\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Let us now incorporate over-sampling by p vectors (say $p = 5$ or $p = 10$):

1. Draw an $n \times (k+p)$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k+p)`
2. Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `Y = A * Omega`
3. Form an $m \times (k+p)$ orthonormal matrix $\mathbf{Q}$ such that $\mathrm{ran}(\mathbf{Q}) = \mathrm{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

Task: Let $\mathbf{A}$ be an $m \times n$ matrix of approximately low rank. We seek to build an approximate ON basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for its range. The objective is to make $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$ small where $\mathbf{Q} = \begin{bmatrix}\mathbf{q}_1 & \cdots & \mathbf{q}_k\end{bmatrix}$. The rank k may or may not be known.

Let us now incorporate over-sampling by p vectors (say $p = 5$ or $p = 10$):

1. Draw an $n \times (k+p)$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k+p)`
2. Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `Y = A * Omega`
3. Form an $m \times (k+p)$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

Important: It turns out to be easy to "filter out" any excess information gathered as a result of over-sampling:

4. Form the $(k+p) \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Form the full SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,'econ')`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

We refer to this full algorithm as the *randomized SVD (RSVD)*.

(Optionally, the factorization can be truncated to drop the last p singular modes.)

Key point: Over-sampling is entirely safe in this context!

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(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 $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Observation: Step (4) contributes an additional step of power iteration.

As a consequence, the leading singular values are computed more accurately than the previous theorem might indicate.

Additionally, you get extra accuracy in the *right* singular vectors:

Theorem: (Y. Dong, P.G. Martinsson, Y. Nakatsukasa 2023) For a rank-l randomized SVD with the Gaussian embedding and $q \geq 0$ power iterations, when the oversampled rank l satisfies $l = \Omega(k)$ (where k is the target rank, $k < l < r = \operatorname{rank}(\mathbf{A})$) and q is reasonably small such that $\eta = \frac{\left(\sum_{j=k+1}^{r} \sigma_j^{4q+2}\right)^2}{\sum_{j=k+1}^{r} \sigma_j^{2(4q+2)}}$ satisfies $\eta = \Omega(l)$, with high probability (at least $1 - e^{-\Theta(k)} - e^{-\Theta(l)}$), there exist distortion factors $0 < \epsilon_1, \epsilon_2 < 1$ such that

(1)
$$\sin \angle_i \left(\mathbf{U}_k, \widehat{\mathbf{U}}_l\right) \leq \left(1 + \frac{1-\epsilon_1}{1+\epsilon_2} \cdot \frac{l}{\sum_{j=k+1}^{r} \sigma_j^{4q+2}} \cdot \sigma_i^{4q+2}\right)^{-\frac{1}{2}}$$

(2)
$$\sin \angle_i \left(\mathbf{V}_k, \widehat{\mathbf{V}}_l\right) \leq \left(1 + \frac{1-\epsilon_1}{1+\epsilon_2} \cdot \frac{l}{\sum_{j=k+1}^{r} \sigma_j^{4q+4}} \cdot \sigma_i^{4q+4}\right)^{-\frac{1}{2}}$$

for all $i \in [k]$, where $\epsilon_1 = \Theta\left(\sqrt{\frac{k}{l}}\right)$ and $\epsilon_2 = \Theta\left(\sqrt{\frac{l}{\eta}}\right)$.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k + p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k + p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k + p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Practical note: The algorithm described is very *communication efficient*, as the interaction with $\mathbf{A}$ is through the matrix-matrix multiplication only.

- High practical speed: matrix-matrix multiplication is very parallelizeable, and has been highly optimized for many architectures.
- Order of magnitude acceleration for data stored out-of-core.
- Highly efficient for GPU computing, or mobile computing (phones, etc).

Note: Krylov methods share the advantage that the interaction with $\mathbf{A}$ is only via the action of the matrix on vectors. The difference is that in the RSVD, the matrix-vector multiplies happen in batches, which is much more efficient.

Block Krylov methods are more similar in this regard.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\boldsymbol{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Question: What if you do not know k in advance?

A theoretician's solution: Try $k = 1$ first, then $k = 2$, $k = 4$, $k = 8$, ...

The simplest solution: Execute the "randomized Gram-Schmidt" process incrementally, one vector at a time. It turns out to be remarkably simple to obtain an estimate for the residual via randomized norm estimation (it is basically free!). In fact,

$$\mathbb{E}\big[\|\mathbf{y}'_{j+1}\|^2\big] = \|\mathbf{A} - \mathbf{Q}_j\mathbf{Q}_j^*\mathbf{A}\|_{\mathrm{F}}^2.$$

This follows from the basic fact that if $\mathbf{g}$ is a Gaussian vector, then $\mathbb{E}\big[\|\mathbf{X}\mathbf{g}\|^2\big] = \|\mathbf{X}\|_{\mathrm{F}}^2$.

Doing it *one block* at a time is even more efficient, and brings Bernstein to bear.

*"randQB" method [SISC **38**(5), pp. S485–S507, 2016]*

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k + p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k + p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k + p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Question: What if you do not know k in advance?

A practical solution: Fix a block size b (say $b = 50$), then incrementally build a factorization of rank $k = b$, $k = 2b$, $k = 3b$, $\dots$, reusing previously computed information.

The matrix $\mathbf{A}$ is driven to upper triangular form through a sequence of unitary maps.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

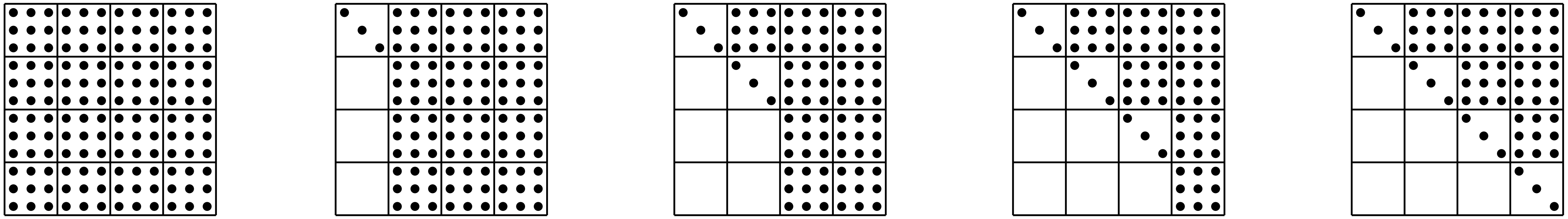
(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Question: What if you do not know k in advance?

A practical solution: Fix a block size b (say $b = 50$), then incrementally build a factorization of rank $k = b$, $k = 2b$, $k = 3b$, $\dots$, reusing previously computed information.

The matrix $\mathbf{A}$ is driven to upper triangular form through a sequence of unitary maps.



$\mathbf{A}_0 = \mathbf{A}$ $\quad$ $\mathbf{A}_1 = \mathbf{U}_1^*\mathbf{A}_0\mathbf{V}_1$ $\quad$ $\mathbf{A}_2 = \mathbf{U}_2^*\mathbf{A}_1\mathbf{V}_2$ $\quad$ $\mathbf{A}_3 = \mathbf{U}_3^*\mathbf{A}_2\mathbf{V}_3$ $\quad$ $\mathbf{A}_4 = \mathbf{U}_4^*\mathbf{A}_3\mathbf{V}_4$

Both $\mathbf{U}_j$ and $\mathbf{V}_j$ are (mostly...) products of b Householder reflectors determined using the randomized range finder. Oversampling is a must, powering is possible.

“Update-free” version has been developed for sparse matrices.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^{*}$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

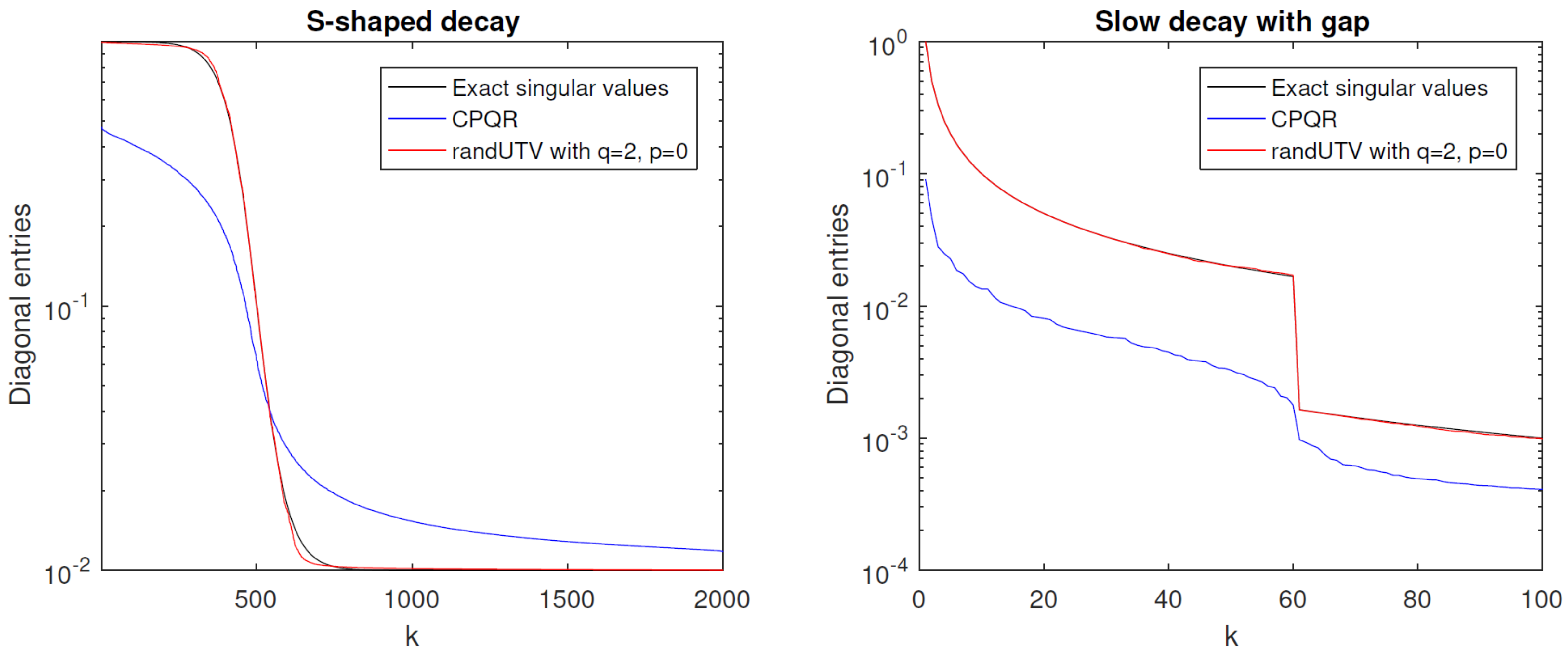
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^{*}\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^{*}\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^{*}$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Numerical results: Exact singular values of $\mathbf{A}$ *vs. diagonal values of* $\mathbf{T}$ *(in* $\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{V}^{*}$*).*



Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k + p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k + p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k + p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Randomized SVD:

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

References:

- C. Chen, N. Heavner, A. Gopal, P.G. Martinsson, “Efficient algorithms for computing rank-revealing factorizations on a GPU”. *Numerical Linear Algebra with Applications*, accepted for publication.
- N. Heavner, P.-G. Martinsson, G. Quintana-Orti, “Computing rank-revealing factorizations of matrices stored out-of-core”, *Concurrency and Computation: Practice and Experience*, in press.
- N. Heavner, F. Igual, G. Quintana-Orti, P.G. Martinsson, “Algorithm 1022: Efficient Algorithms for Computing a Rank-Revealing UTV Factorization on Parallel Computing Architectures”. *ACM TOMS*, **48**(2), pp. 1–42, 2022.
- P.G. Martinsson, G. Quintana-Orti, N. Heavner, “randUTV: A blocked randomized algorithm for computing a rank-revealing UTV factorization.” *ACM TOMS*, **45**(1), pp. 1–26, 2019.

Randomized SVD: Double-sided approximation

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).

Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.

(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.

(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.

(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.

(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Recall that the output of the RSVD is the matrix

$$\mathbf{A}_{\mathrm{approx}} = \mathbf{U}\mathbf{D}\mathbf{V}^* = \mathbf{Q}\mathbf{Q}^*\mathbf{A},$$

where $\mathbf{Q}$ holds the basis for ran($\mathbf{A}$) from the randomized range finder in steps (1)–(3).

Since $\mathbf{Q}\mathbf{Q}^*$ is the orthogonal projector onto ran($\mathbf{A}\mathbf{\Omega}$), we can rewrite $\mathbf{A}_{\mathrm{approx}}$ as:

$$\mathbf{A}_{\mathrm{approx}} = \mathbf{Q}\mathbf{Q}^*\mathbf{A} = (\mathbf{A}\mathbf{\Omega})\,(\mathbf{A}\mathbf{\Omega})^{\dagger}\mathbf{A}.$$

Question: Can we sketch $\mathbf{A}$ from *both sides?*

Randomized SVD: Double-sided approximation

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p (say $p = 5$).
Output: Rank-$(k+p)$ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(1) Draw an $n \times (k+p)$ random matrix $\mathbf{\Omega}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.

(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*$.
(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Recall that the output of the RSVD is the matrix

$$\mathbf{A}_{\rm approx} = \mathbf{U}\mathbf{D}\mathbf{V}^* = \mathbf{Q}\mathbf{Q}^*\mathbf{A},$$

where $\mathbf{Q}$ holds the basis for ran($\mathbf{A}$) from the randomized range finder in steps (1)–(3).

Since $\mathbf{Q}\mathbf{Q}^*$ is the orthogonal projector onto ran($\mathbf{A}\mathbf{\Omega}$), we can rewrite $\mathbf{A}_{\rm approx}$ as:

$$\mathbf{A}_{\rm approx} = \mathbf{Q}\mathbf{Q}^*\mathbf{A} = (\mathbf{A}\mathbf{\Omega})\,(\mathbf{A}\mathbf{\Omega})^{\dagger}\mathbf{A}.$$

Question: Can we sketch $\mathbf{A}$ from *both sides?*

Answer: Yes; simply draw a sketch $\mathbf{\Psi}\mathbf{A}$ of the row space as well. Then

$$\mathbf{A}_{\rm newapprox} = (\mathbf{A}\mathbf{\Omega})\,(\mathbf{A}\mathbf{\Omega})^{\dagger}\,\mathbf{A}\,(\mathbf{\Psi}\mathbf{A})^{\dagger}\,(\mathbf{\Psi}\mathbf{A}) = \cdots = (\mathbf{A}\mathbf{\Omega})\,(\mathbf{\Psi}\mathbf{A}\mathbf{\Omega})^{\dagger}\,(\mathbf{\Psi}\mathbf{A}).$$

(Formula goes back at least to Wedderburn 1934.)

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ and $\mathbf{\Psi} \in \mathbb{R}^{(k+p')\times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A}\mathbf{\Omega})\left(\mathbf{\Psi}\mathbf{A}\mathbf{\Omega}\right)^{\dagger}(\mathbf{\Psi}\mathbf{A}) =: \mathbf{A}_{\rm approx}$.

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ and $\mathbf{\Psi} \in \mathbb{R}^{(k+p')\times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A\Omega})\left(\mathbf{\Psi A \Omega}\right)^{\dagger}(\mathbf{\Psi A}) =: \mathbf{A}_{\mathrm{approx}}$.

Observation 1:

The matrix $\mathbf{A}_{\mathrm{approx}}$ can be built *in a single pass over* $\mathbf{A}$.

In other words, we need to view each matrix entry of $\mathbf{A}$ only once.

This cannot be done using deterministic methods (as far as I know).

“Streaming” or “single-view” algorithm.

Note: *Using different over-sampling parameters p and p' for the column and row spaces is often better.*

References: Alon, Gibbons, Matias and Szegedy (2002); Woolfe, Liberty, Rokhlin, and Tygert (2008); Clarkson and Woodruff (2009); Li, Nguyen and Woodruff (2014); Boutsidi, Woodruff and Zhong (2016), Tropp, Yurtsever, Udell and Cevher (2017); Pourkamali-Anaraki and Becker (2019); Wang, Gittens and Mahoney (2019); Nakatsukasa, *arXiv:2009.11392*, 2020; Dong & Martinsson, *arXiv:2104.05877*, 2021; …many more …

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\boldsymbol{\Omega} \in \mathbb{R}^{m \times (k+p)}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{(k+p') \times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A}\boldsymbol{\Omega}) \left(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}\right)^{\dagger} (\boldsymbol{\Psi}\mathbf{A}) =: \mathbf{A}_{\text{approx}}$.

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A}\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}\mathbf{A}$ requires $O(mnk)$ flops.

$O(mnk)$ matches the flop count of Gram-Schmidt, or a Krylov method applied to a dense matrix.

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ and $\mathbf{\Psi} \in \mathbb{R}^{(k+p')\times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A\Omega})\left(\mathbf{\Psi A \Omega}\right)^{\dagger}(\mathbf{\Psi A}) =: \mathbf{A}_{\rm approx}$.

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A\Omega}$ and $\mathbf{\Psi A}$ requires $O(mnk)$ flops.

Instead of Gaussian random matrices, we can use *structured random matrices* $\mathbf{\Psi}$ and $\mathbf{\Omega}$ with the property that $\mathbf{A\Omega}$ *and* $\mathbf{\Psi A}$ *can be evaluated using asymptotically fewer flops.*

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ and $\mathbf{\Psi} \in \mathbb{R}^{(k+p')\times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A\Omega})\,(\mathbf{\Psi A\Omega})^{\dagger}\,(\mathbf{\Psi A}) =: \mathbf{A}_{\rm approx}$.

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A\Omega}$ and $\mathbf{\Psi A}$ requires $O(mnk)$ flops.

Instead of Gaussian random matrices, we can use *structured random matrices* $\mathbf{\Psi}$ and $\mathbf{\Omega}$ with the property that $\mathbf{A\Omega}$ *and* $\mathbf{\Psi A}$ *can be evaluated using asymptotically fewer flops.*

- 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 sign matrix”. Place r random entries in each row of $\mathbf{\Omega}$. (Say $r = 2$ or $r = 4$.) Cost is now $O(mn)$!

Randor

Task: F

Algoritl n.

Form ap

Observ

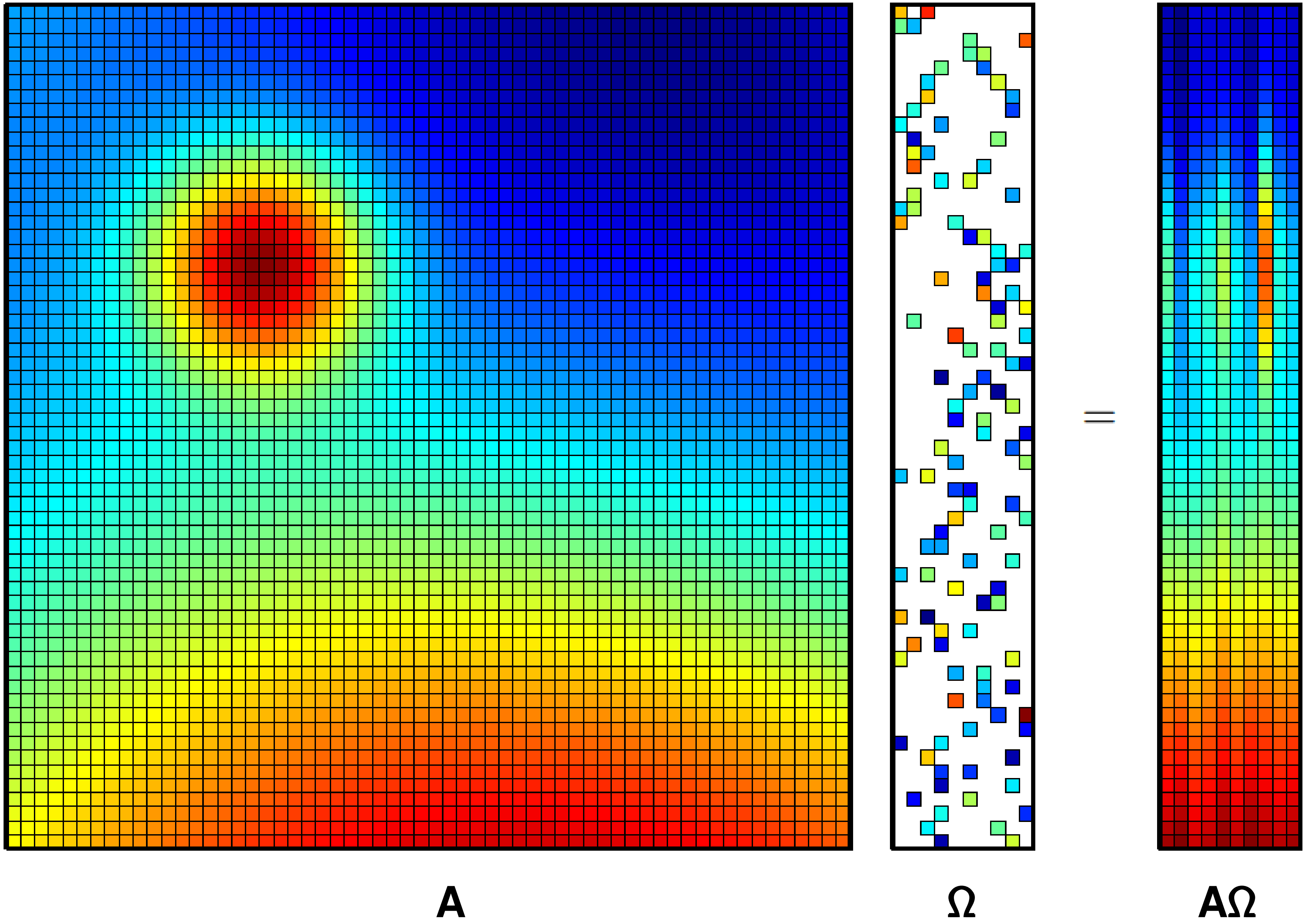
Using G ps.

Instead s $\mathbf{\Psi}$ and $\mathbf{\Omega}$

with the *r flops.*

- Rar $n\log(k)$).
- Cha
- “Spa or $r = 4$.)

Cos



The matrix $\mathbf{\Omega}$ *is a sparse random matrix. Two nonzero entries are placed randomly in each row. In consequence, each column of* $\mathbf{A}$ *contributes to precisely two columns of the sample matrix* $\mathbf{Y} = \mathbf{A\Omega}$*. This structured random map has* $O(mn)$ *complexity, is easy to work with practically, and often provides good accuracy.*

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ and $\mathbf{\Psi} \in \mathbb{R}^{(k+p')\times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A\Omega})\left(\mathbf{\Psi A\Omega}\right)^{\dagger}(\mathbf{\Psi A}) =: \mathbf{A}_{\rm approx}$.

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A\Omega}$ and $\mathbf{\Psi A}$ requires $O(mnk)$ flops.

Instead of Gaussian random matrices, we can use *structured random matrices* $\mathbf{\Psi}$ and $\mathbf{\Omega}$ with the property that $\mathbf{A\Omega}$ *and* $\mathbf{\Psi A}$ *can be evaluated using asymptotically fewer flops.*

- 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 sign matrix". Place r random entries in each row of $\mathbf{\Omega}$. (Say $r = 2$ or $r = 4$.) Cost is now $O(mn)$!

When a structured random matrix is used, overall cost can be reduced to $O(mn + k^3)$.

Despite the pseudo-inverse, this can be done in a numerically stable way.

Randomized SVD: Double-sided approximation

Task: Find a rank-k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Algorithm: Draw Gaussian random matrices $\boldsymbol{\Omega} \in \mathbb{R}^{m \times (k+p)}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{(k+p') \times n}$.

Form approximation $\mathbf{A} \approx (\mathbf{A}\boldsymbol{\Omega}) \left(\boldsymbol{\Psi}\mathbf{A}\boldsymbol{\Omega}\right)^{\dagger} (\boldsymbol{\Psi}\mathbf{A}) =: \mathbf{A}_{\mathrm{approx}}$.

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A}\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}\mathbf{A}$ requires $O(mnk)$ flops.

Instead of Gaussian random matrices, we can use *structured random matrices* $\boldsymbol{\Psi}$ and $\boldsymbol{\Omega}$ with the property that $\mathbf{A}\boldsymbol{\Omega}$ *and* $\boldsymbol{\Psi}\mathbf{A}$ *can be evaluated using asymptotically fewer flops.*

- 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 sign matrix". Place r random entries in each row of $\boldsymbol{\Omega}$. (Say $r = 2$ or $r = 4$.) Cost is now $O(mn)$!

Theory is very weak, and typically requires $\ell = k + p \sim k\log(k)$.

In practice, $\ell \approx 2k$ is almost always more than enough.

References: Ailon & Chazelle (2006); Liberty, Rokhlin, Tygert, and Woolfe (2006); Halko, Martinsson, Tropp (2011); Clarkson & Woodruff (2013); Meng & Mahoney (2013); Nelson & Nguyen (2013); Urano (2013); Nakatsukasa, *arXiv:2009.11392*, 2020; Dong & Martinsson, *arXiv:2104.05877*, 2021. Much subsequent work — "Fast Johnson-Lindenstrauss transform."

Randomized SVD: A posteriori error estimation

With Yuji Nakatsukasa, 2024

Question: How estimate the error in a specific instantiation of a randomized algorithm?

Our framing is that we are given an $m \times n$ matrix $\mathbf{A}$, and use the randomized range finder to build an approximation:

- Draw an $n \times \ell$ random matrix $\mathbf{\Omega}$, for say $\ell = k + 10$ or $\ell = 2k$.

  Note: We may use a very fast, but possibly less reliable, class of random matrices.
- Form $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, and perform QR factorization $[\mathbf{Q}, \sim] = \mathtt{qr}(\mathbf{Y}, 0)$.
- Use $\mathbf{A}_{\rm approx} = \mathbf{Q}\left(\mathbf{Q}^*\mathbf{A}\right)$ as the approximation.

Our objective is to estimate the error

$$e = \|\mathbf{A} - \mathbf{A}_{\rm approx}\|$$

using only the information at hand!

Why care?

- Enable the use of “risky” random dimension reducing maps.

  Cf. the “responsibly reckless” mode of computing, in Dongarra’s terminology.
- The numerical rank of $\mathbf{A}$ may not be known in advance.
- Need bounds and estimates that involve *only quantities actually at hand.*

Randomized SVD: A posteriori error estimation

With Yuji Nakatsukasa, 2024

Question: How estimate the error in a specific instantiation of a randomized algorithm?

Answer: You can very inexpensively compute a *certificate of accuracy*. To illustrate, consider the RSVD: We draw a random matrix $\mathbf{\Omega}$, set $\mathbf{Y} = \mathbf{A\Omega}$, and approximate $\mathbf{A}$ via

$$\mathbf{A}_{\rm approx} = \mathbf{QQ}^*\mathbf{A},$$

where $[\mathbf{Q}, \sim] = \mathrm{qr}(\mathbf{A}, 0)$. We seek to estimate the error

$$e = \|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}.$$

The idea is to draw a thin slice of a Gaussian, $\mathbf{G} \in \mathbb{R}^{n\times q}$, for $q = 10$ say, that is quarantined from the construction of $\mathbf{A}_{\rm approx}$ and is used purely to estimate the error. Evaluate $\mathbf{B} = \mathbf{AG}$. Then use that for any matrix $\mathbf{H}$, we have

$$\mathbb{E}\big[\|\mathbf{HG}\|_{\rm F}^2\big] = q\|\mathbf{H}\|_{\rm F}^2.$$

Setting $\mathbf{H} = \mathbf{A} - \mathbf{A}_{\rm approx}$, we use the estimate

$$\|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}^2 \approx \frac{1}{q}\|(\mathbf{A} - \mathbf{A}_{\rm approx})\mathbf{G}\|_{\rm F}^2 = \frac{1}{q}\|(\mathbf{I} - \mathbf{QQ}^*)\mathbf{AG}\|_{\rm F}^2 = \frac{1}{q}\|(\mathbf{I} - \mathbf{QQ}^*)\mathbf{B}\|_{\rm F}^2.$$

Very reliable. Inexpensive.

Martinsson, Tropp, Acta Numerica 2020, Sec. 12.2.

But we can do better still!

New!

Randomized SVD: A posteriori error estimation

With Yuji Nakatsukasa, 2024

Question: How estimate the error in a specific instantiation of a randomized algorithm?

Recall from previous slide: Draw a random matrix $\mathbf{\Omega}$, set $\mathbf{Y} = \mathbf{A\Omega}$, approximate $\mathbf{A}$ via $\mathbf{A}_{\rm approx} = \mathbf{QQ}^*\mathbf{A}$, where $[\mathbf{Q}, \sim] = \texttt{qr}(\mathbf{Y}, 0)$. We seek to estimate $e = \|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}$. To get a *certificate of accuracy*, we draw $\mathbf{G}$, set $\mathbf{B} = \mathbf{AG}$, and use $e^2 \approx \frac{1}{q}\|(\mathbf{I} - \mathbf{QQ}^*)\mathbf{B}\|_{\rm F}^2$.

Randomized SVD: A posteriori error estimation

With Yuji Nakatsukasa, 2024

Question: How estimate the error in a specific instantiation of a randomized algorithm?

Recall from previous slide: Draw a random matrix $\mathbf{\Omega}$, set $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, approximate $\mathbf{A}$ via $\mathbf{A}_{\rm approx} = \mathbf{Q}\mathbf{Q}^*\mathbf{A}$, where $[\mathbf{Q},\sim] = \mathtt{qr}(\mathbf{Y},0)$. We seek to estimate $e = \|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}$. To get a *certificate of accuracy*, we draw $\mathbf{G}$, set $\mathbf{B} = \mathbf{A}\mathbf{G}$, and use $e^2 \approx \frac{1}{q}\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2$.

Acceleration via fast sketching: Observe that $\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2$ is the minimal error when seeking to fit $\mathbf{B}$ in $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{A}\mathbf{\Omega}) = \text{ran}(\mathbf{Y})$. So

$$\|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}^2 \approx \frac{1}{q}\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2 = \frac{1}{q}\inf_{\mathbf{M}\in\mathbb{R}^{\ell\times q}}\|\mathbf{Y}\mathbf{M} - \mathbf{B}\|_{\rm F}^2.$$

Next, we *sketch* the least squares problem. Draw FJLT $\mathbf{\Psi} \in \mathbb{R}^{m\times s}$, with $s = O(k)$, then

$$\|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}^2 \approx \frac{1}{q}\frac{m}{s}\inf_{\mathbf{M}\in\mathbb{R}^{\ell\times q}}\|\mathbf{\Psi}^*(\mathbf{Y}\mathbf{M} - \mathbf{B})\|_{\rm F}^2.$$

So in the end, all we need to do is to approximately solve the least squares problem

$$\begin{array}{ccccc} (\mathbf{\Psi}^*\mathbf{Y}) & \mathbf{M} & = & (\mathbf{\Psi}^*\mathbf{B}) \\ s\times\ell & \ell\times q & & s\times q \end{array}$$

The total extra cost (beyond the cost of RSVD):

- q extra matvecs. (Think $q = 10$.) Can be done as a block.
- Sketching step – evaluate $\mathbf{\Psi}^*\mathbf{Y}$ and $\mathbf{\Psi}^*\mathbf{B}$. Cost $O(m\ell)$.
- Solve small least squares problem. Cost $O(s\,\ell^2) = O(k^3)$.

Randomized SVD: A posteriori error estimation

With Yuji Nakatsukasa, 2024

Question: How estimate the error in a specific instantiation of a randomized algorithm?

Recall from previous slide: Draw a random matrix $\mathbf{\Omega}$, set $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, approximate $\mathbf{A}$ via $\mathbf{A}_{\rm approx} = \mathbf{Q}\mathbf{Q}^*\mathbf{A}$, where $[\mathbf{Q}, \sim] = \texttt{qr}(\mathbf{Y}, 0)$. We seek to estimate $e = \|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}$. To get a *certificate of accuracy*, we draw $\mathbf{G}$, set $\mathbf{B} = \mathbf{A}\mathbf{G}$, and use $e^2 \approx \frac{1}{q}\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2$.

Acceleration via fast sketching: Observe that $\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2$ is the minimal error when seeking to fit $\mathbf{B}$ in ran$(\mathbf{Q})$ = ran$(\mathbf{A}\mathbf{\Omega})$ = ran$(\mathbf{Y})$. So

$$\|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}^2 \approx \frac{1}{q}\|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{B}\|_{\rm F}^2 = \frac{1}{q}\inf_{\mathbf{M}\in\mathbb{R}^{\ell\times q}}\|\mathbf{Y}\mathbf{M} - \mathbf{B}\|_{\rm F}^2.$$

Next, we *sketch* the least squares problem. Draw FJLT $\mathbf{\Psi} \in \mathbb{R}^{m\times s}$, with $s = O(k)$, then

$$\|\mathbf{A} - \mathbf{A}_{\rm approx}\|_{\rm F}^2 \approx \frac{1}{q}\frac{m}{s}\inf_{\mathbf{M}\in\mathbb{R}^{\ell\times q}}\|\mathbf{\Psi}^*(\mathbf{Y}\mathbf{M} - \mathbf{B})\|_{\rm F}^2.$$

So in the end, all we need to do is to approximately solve the least squares problem

$$\begin{array}{cccc} (\mathbf{\Psi}^*\mathbf{Y}) & \mathbf{M} & = & (\mathbf{\Psi}^*\mathbf{B}) \\ s\times\ell & \ell\times q & & s\times q \end{array}$$

In conclusion:

- *Very* cheap error estimation. Gives accurate estimates.
- Involves only available data.
- No need to even form $\mathbf{Q}$! Can be done as soon as $\mathbf{A}\mathbf{\Omega}$ is available.

Numerical experiments

Question: Which type of random matrix should I use for the sketching?

We will compare:

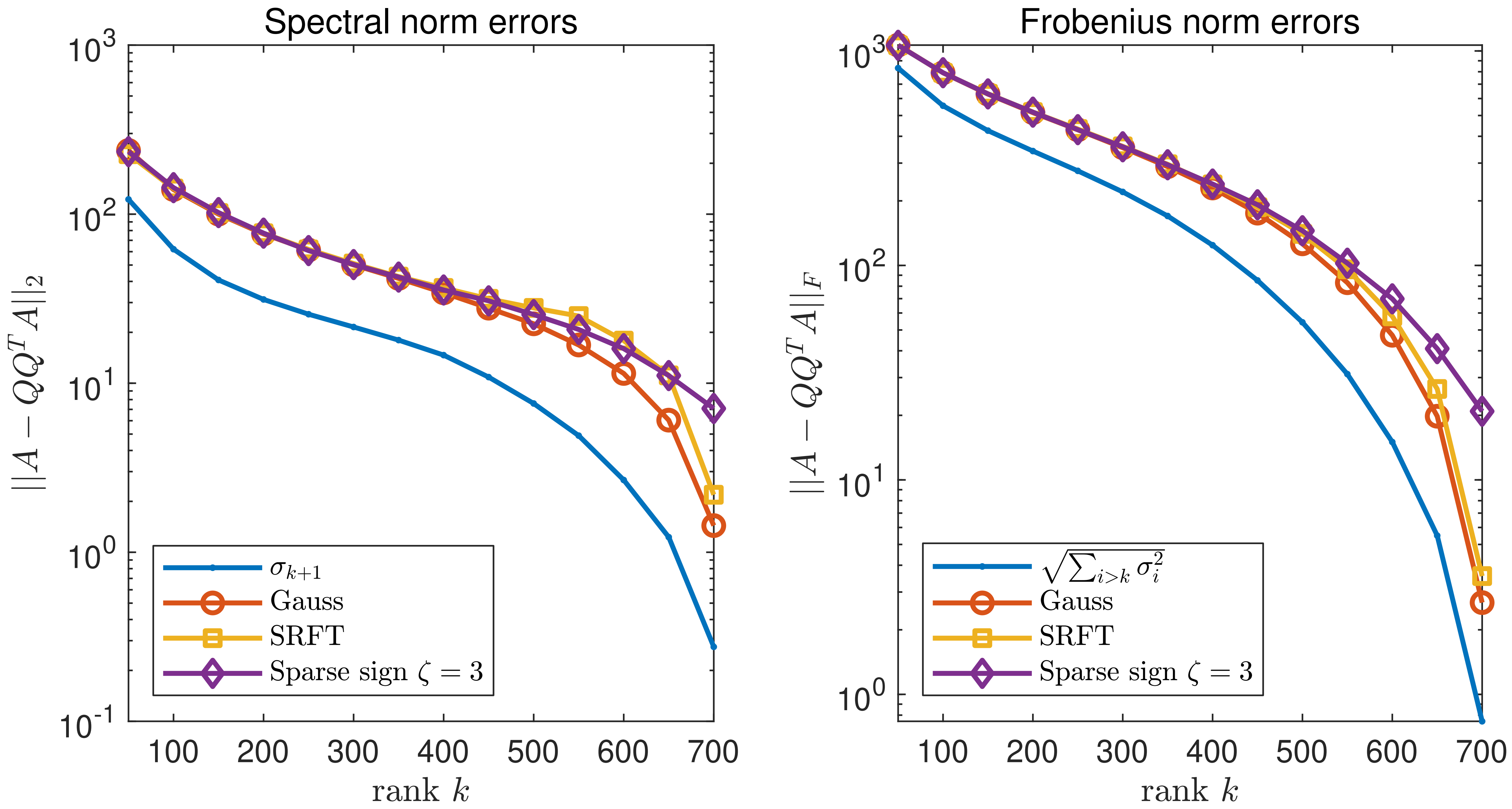
- Optimality: How good of a basis for the row space do you get?
- Computational cost: What is the *practical* speed?

(Dong & Martinsson, to appear in Adv. Comp. Math. Also arXiv:2104.05877)

Comparison of different random matrices — accuracy

Compare picking Ω as (1) Gaussian, (2) SRFT, (3) sparse random.

(To be precise, in these experiments, we sketched the *column space*, not the row space.)

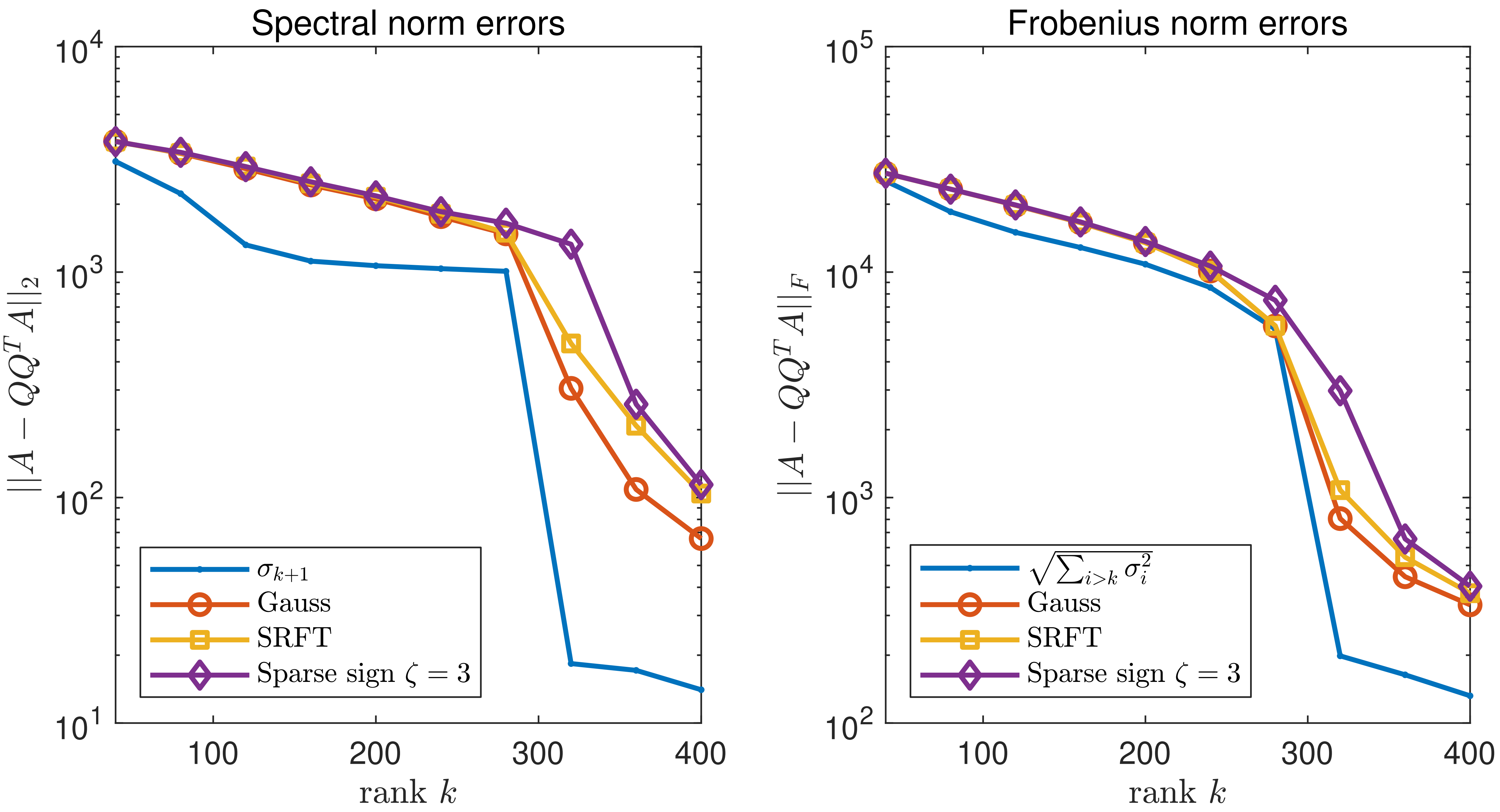


*The "*MNIST*" test matrix is dense and of size* 784 × 60 000 *where each column holds one hard drawn digit between 0 and 9. The matrix is* 80% *sparse.*

Comparison of different random matrices — accuracy

Compare picking Ω as (1) Gaussian, (2) SRFT, (3) sparse random.

(To be precise, in these experiments, we sketched the *column space*, not the row space.)

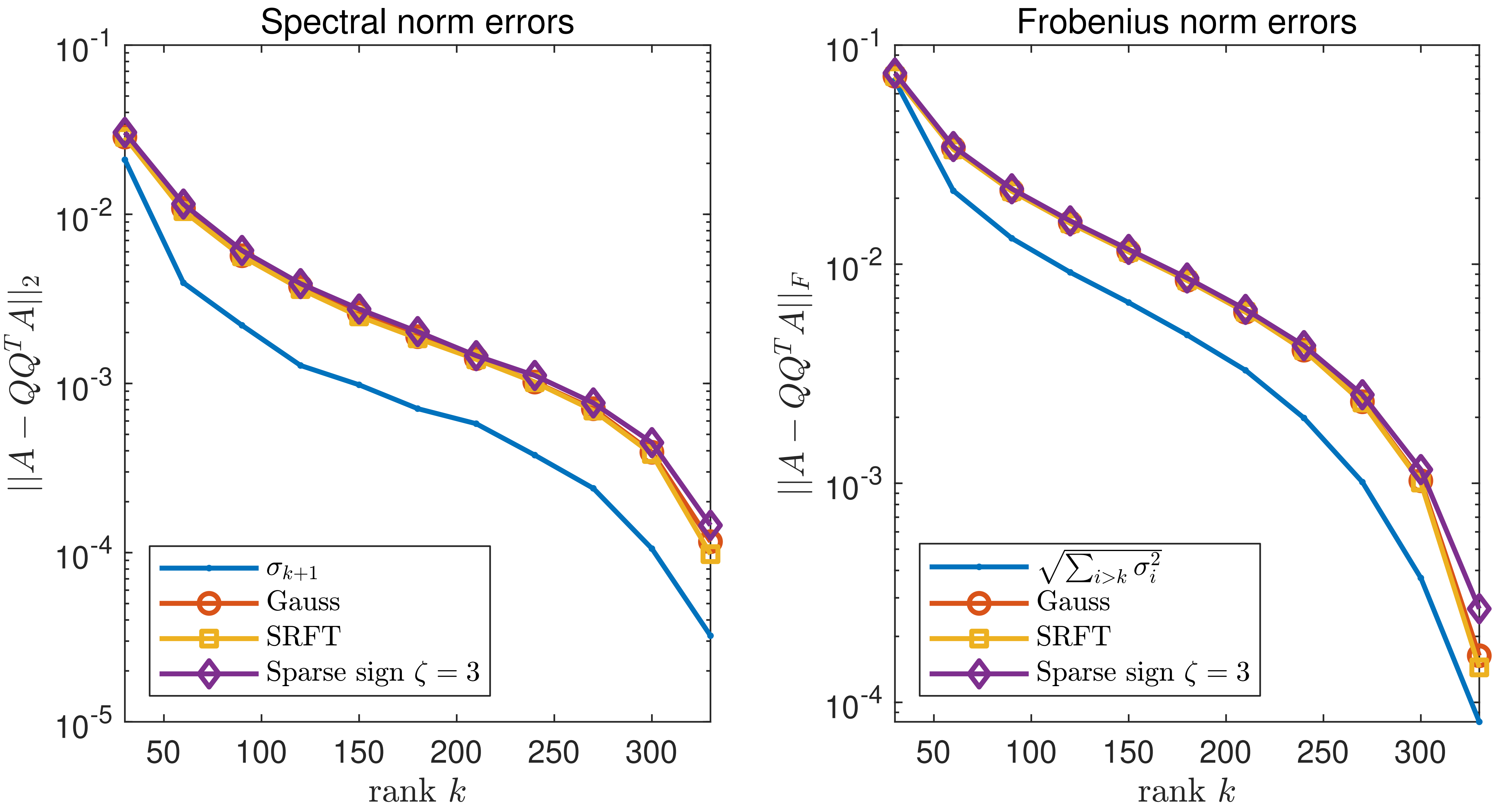


The "LARGE" test matrix is taken from a linear programming example. It is sparse, of size 4 282 × 8 617*, with* 20 635 *nonzero entries.*

Comparison of different random matrices — accuracy

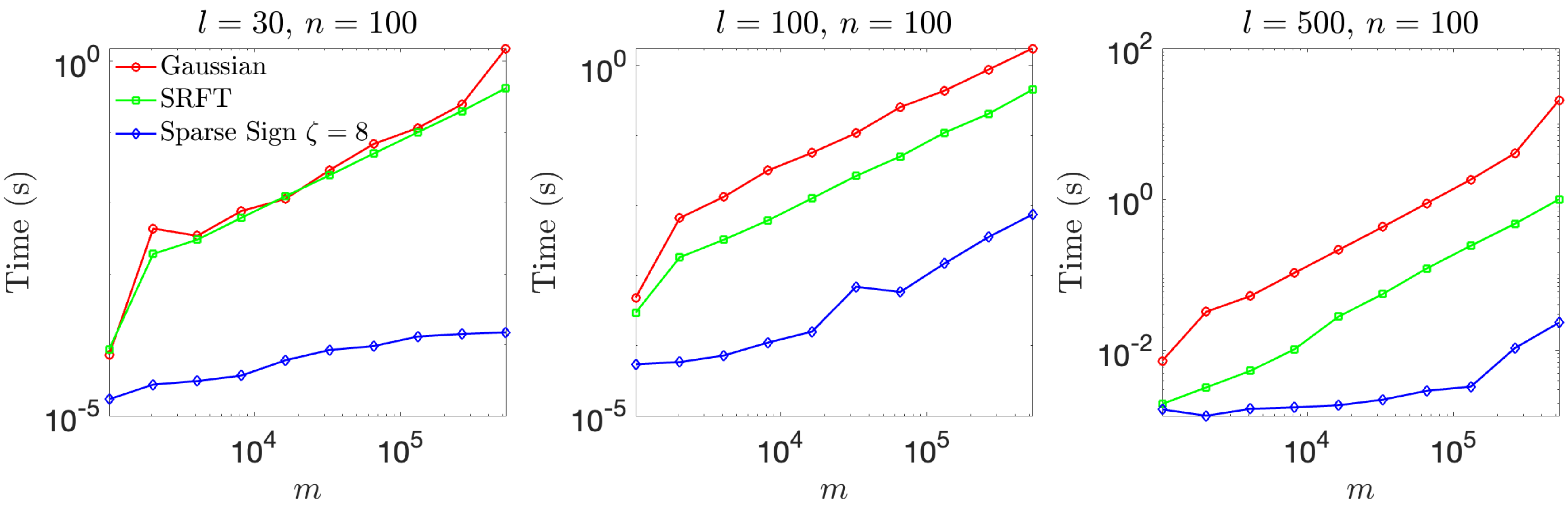
Compare picking Ω as (1) Gaussian, (2) SRFT, (3) sparse random.

(To be precise, in these experiments, we sketched the *column space*, not the row space.)



The "SNN" test matrix has been used in the CUR literature before. It is an artificial sparse matrix of size 1 000 × 1 000.

Comparison of different random matrices — execution time



The runtime of applying different subspace embeddings $\mathbf{\Omega} \in \mathbb{R}^{\ell \times m}$ to an arbitrary dense matrix of size $m \times n$, scaled with respect to the ambient dimension m, at different embedding dimensions ℓ and a fixed number of columns $n = 100$.

(Note: This n is artificially small, but the scaling with n is linear.)

Note: Observe that the dimension of the sketch is quite high in these examples.

Random mixing for solving linear systems & least squares problems

Having access to *unitary* random maps opens the door other linear algebraic problems:

Computing *full* rank-revealing factorizations: Let $\mathbf{A}$ be an $n \times n$ matrix. To build a *full* rank-revealing factorization of $\mathbf{A}$, draw $\boldsymbol{\Omega}$ from a Haar distribution, and form

$$\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}.$$

Then perform a QR factorization of $\mathbf{Y}$, so that

$$\mathbf{Y} = \mathbf{Q}\mathbf{R}.$$

One can prove that then a "rank-revealing" factorization is obtained through

$$\mathbf{A} = \mathbf{Q}\mathbf{R}\boldsymbol{\Omega}^*.$$

The point here is communication efficiency. *(Demmel, Dumitriu, Holtz 2007)*

Setting $\mathbf{Y} = \mathbf{A}\mathbf{A}^*\mathbf{A}\boldsymbol{\Omega}$ is even better! *(Heavner, Chen, Gopal, Martinsson 2023)*

Solving linear systems without pivoting: Consider

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

Randomized preconditioning results in a system that can be solved *without pivoting*

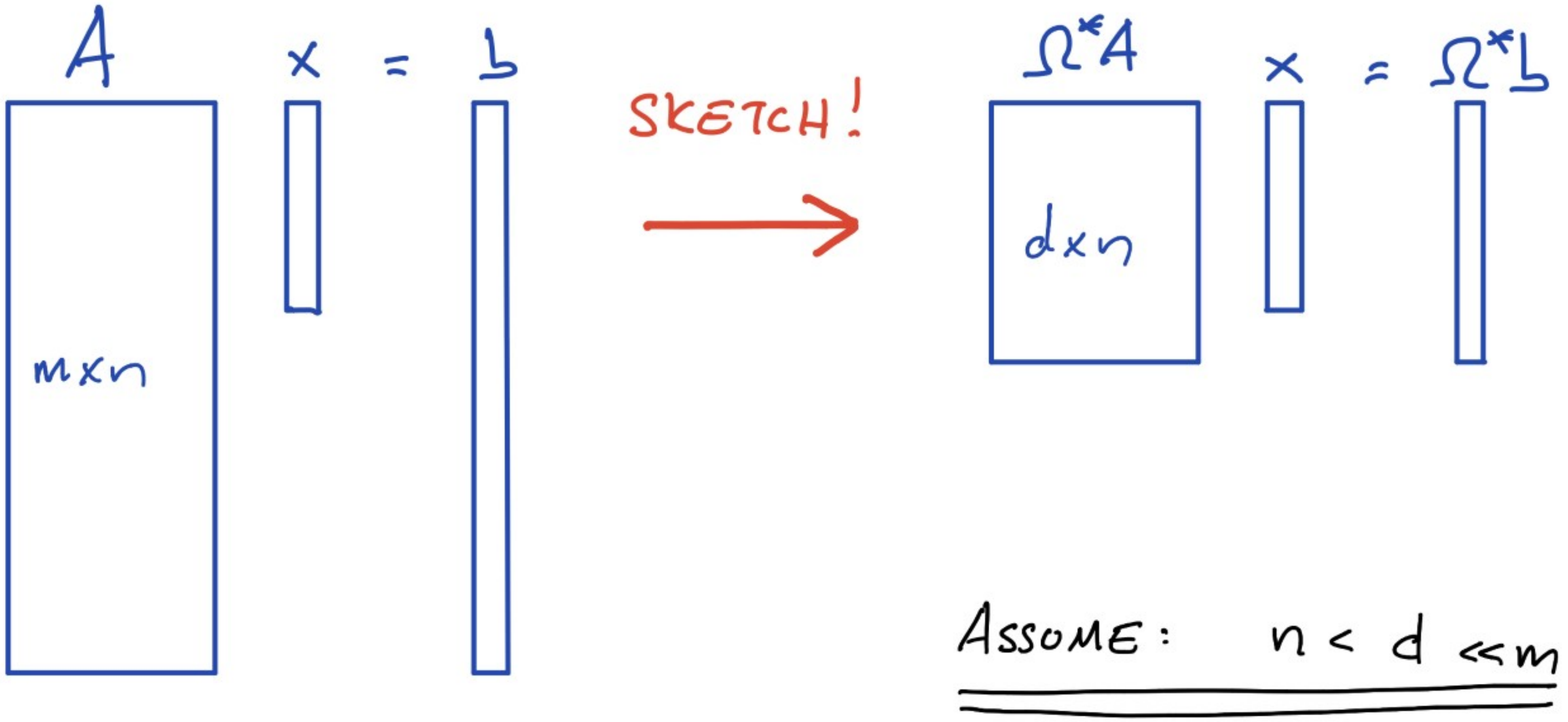
$$\big(\boldsymbol{\Psi}^*\mathbf{A}\boldsymbol{\Omega}\big)\,\big(\boldsymbol{\Omega}^*\mathbf{x}\big) = \boldsymbol{\Psi}^*\mathbf{b}.$$

Again, the point is communication efficiency. *(Parker 1995)*

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 **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 $(\mathbf{AM}^{-1})\,(\mathbf{Mx}) = \mathbf{b}$.

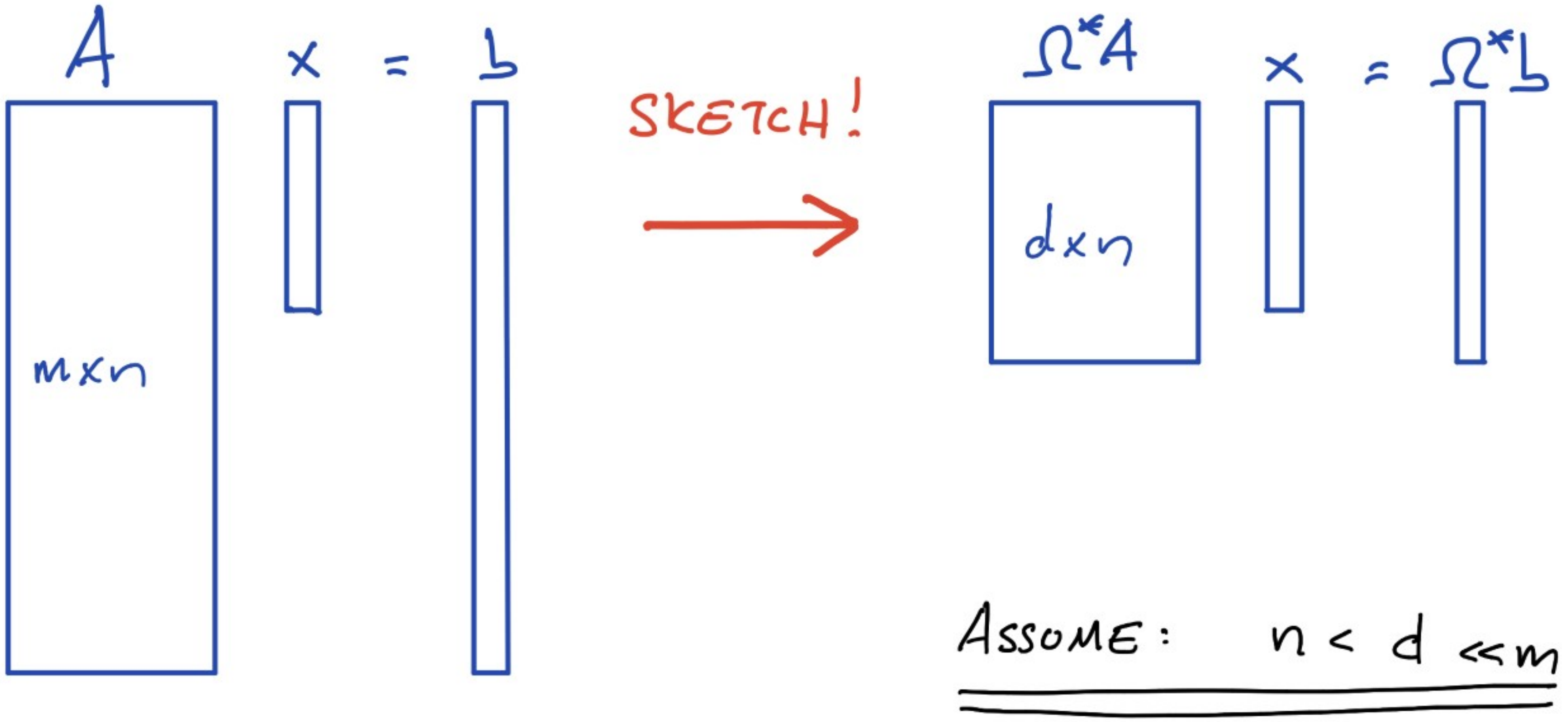
Ideal use of randomization. Guaranteed accuracy if you evaluate the residual.

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.

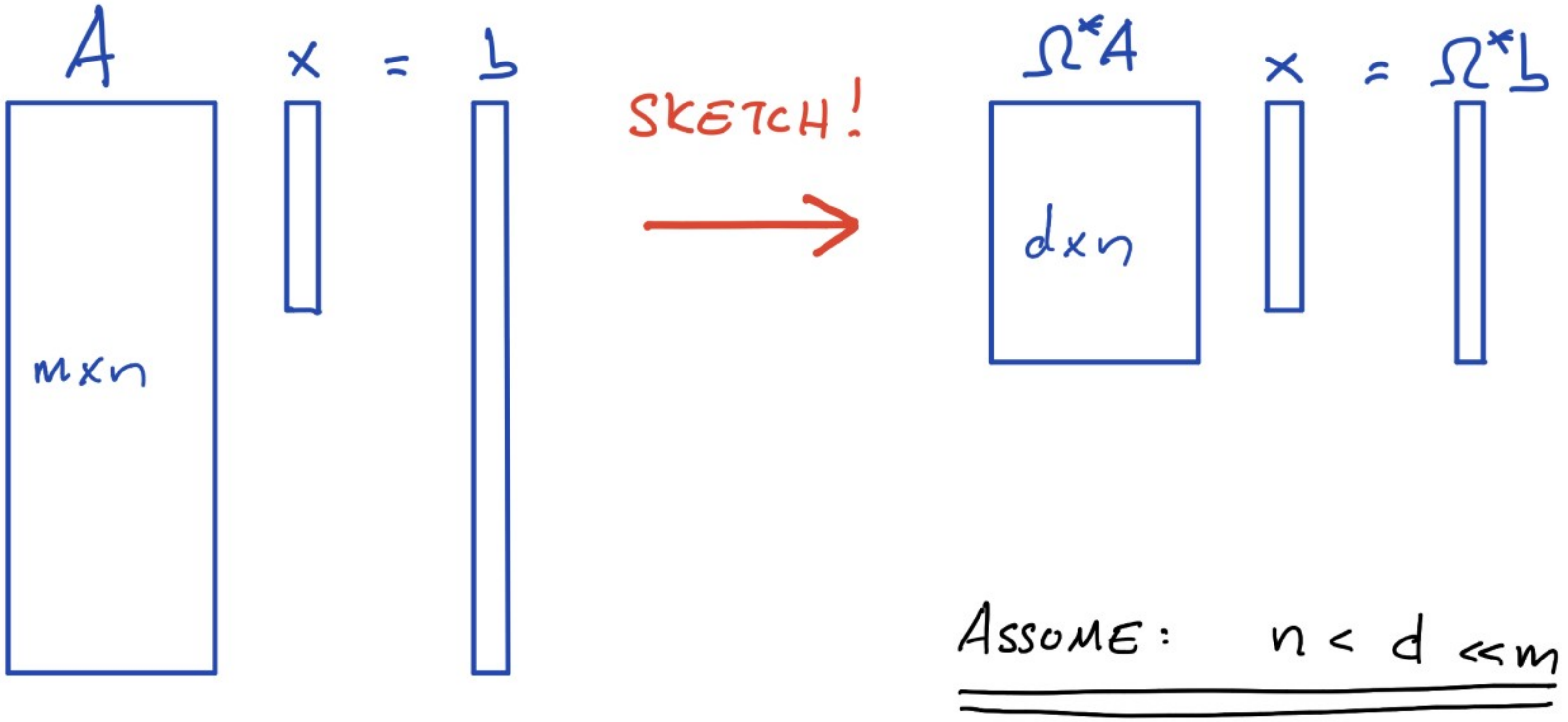


Question: Can the sketch be *down-sampling*? Simply pick *d* equations randomly?

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 **A** has structure, as in tensor approximation.

Outline

- **Methods based on randomized embeddings**
  - Low rank approximation: The randomized SVD.
  - Streaming and single-pass methods.
  - A posteriori error estimation.
  - Structured random maps (“fast Johnson-Lindenstrauss transforms”).
  - Linear solvers: Sketch-to-solve vs. Sketch-to-precondition.
  - Las Vegas style methods: Extremely robust and reliable.

- **Methods based on randomized sampling**
  - Monte Carlo style methods → less reliable, less accurate, less robust.
  - Enable the solution of stupendously large problems that would otherwise be intractable.

- **Approximation of rank-structured matrices**
  - Approximation of global operators of mathematical physics (solution operators, DtNs, . . .).
  - Tools for matrices that are not of global low rank, but have structure that can be exploited.
  - Conceptually related to Fast Multipole Methods, Fast Direct Solvers, Calderón-Zygmund theory . . .

Matrix approximation by sampling

Two paradigms for how to use randomization to in linear algebra:

Randomized embeddings (What we have discussed so far.)	**Randomized sampling** (What we will discuss next.)
Often faster than classical deterministic methods.	Sometimes *far* faster than classical deterministic methods. Faster than matrix-vector multiplication, even.
Highly reliable and robust.	Can fail in the “general” case.
High accuracy is attainable.	Typically low accuracy.
Best for scientific computing.	Enables solution of large scale problems in “big data” where no other methods work.
	Powerful tool in situations where you only have access to “oversampled” data that artificially inflates the dimensionality of the problem.

Matrix approximation by sampling

Suppose that $\mathbf{A} = \sum_{t=1}^{T} \mathbf{A}_t$ where each $\mathbf{A}_t$ is “simple” in some sense.

Matrix approximation by sampling

Suppose that $\mathbf{A} = \sum_{t=1}^{T} \mathbf{A}_t$ where each $\mathbf{A}_t$ is "simple" in some sense.

Example: Sparse matrix written as a sum over its nonzero entries

$$\underbrace{\begin{bmatrix} 5 & -2 & 0 \\ 0 & 0 & -3 \\ 1 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}} = \underbrace{\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}_1} + \underbrace{\begin{bmatrix} 0 & -2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}_2} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}_3} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}_4}$$

Example: Each $\mathbf{A}_i$ could be a column of the matrix

$$\underbrace{\begin{bmatrix} 5 & -2 & 7 \\ 1 & 3 & -3 \\ 1 & -1 & 1 \end{bmatrix}}_{=\mathbf{A}} = \underbrace{\begin{bmatrix} 5 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{=\mathbf{A}_1} + \underbrace{\begin{bmatrix} 0 & -2 & 0 \\ 0 & 3 & 0 \\ 0 & -1 & 0 \end{bmatrix}}_{=\mathbf{A}_2} + \underbrace{\begin{bmatrix} 0 & 0 & 7 \\ 0 & 0 & -3 \\ 0 & 0 & 1 \end{bmatrix}}_{=\mathbf{A}_3}.$$

Example: Matrix-matrix multiplication broken up as a sum of rank-1 matrices:

$$\mathbf{A} = \mathbf{BC} = \sum_{t} \mathbf{B}(:, t)\mathbf{C}(t, :).$$

Matrix approximation by sampling

Suppose that $\mathbf{A} = \sum_{t=1}^{T} \mathbf{A}_t$ where each $\mathbf{A}_t$ is "simple" in some sense.

Let $\{p_t\}_{t=1}^{T}$ be a probability distribution on the index vector $\{1, 2, \ldots, T\}$.

Draw an index $t \in \{1, 2, \ldots, T\}$ according to the probability distribution given, and set

$$\mathbf{X} = \frac{1}{p_t}\mathbf{A}_t.$$

Then from the definition of the expectation, we have

$$\mathbb{E}\left[\mathbf{X}\right] = \sum_{t=1}^{T} p_t \times \frac{1}{p_t}\mathbf{A}_t = \sum_{t=1}^{T} \mathbf{A}_t = \mathbf{A},$$

so $\mathbf{X}$ is an *unbiased estimate* of $\mathbf{A}$.

Clearly, a single draw is not a good approximation — unrepresentative, *large variance*.

Instead, draw several samples and average:

$$\bar{\mathbf{X}} = \frac{1}{k}\sum_{t=1}^{k} \mathbf{X}_t,$$

where $\mathbf{X}_t$ are independent samples from the same distribution.

As k grows, the variance will decrease, as usual. Various Bernstein inequalities apply.

Matrix approximation by sampling

As an illustration of the theory, we cite a matrix-Bernstein result from J. Tropp (2015):

Theorem: Let $\mathbf{A} \in \mathbb{R}^{m\times n}$. Construct a probability distribution for $\mathbf{X} \in \mathbb{R}^{m\times n}$ that satisfies

$$\mathbb{E}\big[\mathbf{X}\big] = \mathbf{A} \quad \text{and} \quad \|\mathbf{X}\| \leq R.$$

Define the per-sample second-moment: $v(\mathbf{X}) := \max\{\|\mathbb{E}[\mathbf{X}\mathbf{X}^*]\|, \|\mathbb{E}[\mathbf{X}^*\mathbf{X}]\|\}$.

Form the matrix sampling estimator: $\bar{\mathbf{X}}_k = \frac{1}{k}\sum_{t=1}^{k} \mathbf{X}_i$ where $\mathbf{X}_t \sim \mathbf{X}$ are iid.

Then $\mathbb{E}\|\bar{\mathbf{X}}_k - \mathbf{A}\| \leq \sqrt{\dfrac{2v(\mathbf{X})\log(m+n)}{k}} + \dfrac{2R\log(m+n)}{3k}$.

Furthermore, for all $s \geq 0$: $\mathbb{P}\big[\|\bar{\mathbf{X}}_k - \mathbf{A}\| \geq s\big] \leq (m+n)\exp\left(\dfrac{-ks^2/2}{v(\mathbf{X}) + 2Rs/3}\right)$.

Suppose that we want $\mathbb{E}\|\mathbf{A} - \bar{\mathbf{X}}\| \leq 2\epsilon$. The theorem says to pick

$$k \geq \max\left\{\frac{2v(\mathbf{X})\log(m+n)}{\epsilon^2}, \frac{2R\log(m+n)}{3\epsilon}\right\}$$

In other words, the number k of samples should be proportional to both $v(\mathbf{X})$ and to the upper bound R.

The scaling $k \sim \dfrac{1}{\epsilon^2}$ is discouraging, and unavoidable (since error $\epsilon \sim 1/\sqrt{k}$).

Matrix approximation by sampling: Matrix matrix multiplication

Given two matrices **B** and **C**, consider the task of evaluating

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times T}{\mathbf{B}} \ \underset{T\times n}{\mathbf{C}} = \sum_{t=1}^{T} \mathbf{B}(:,t)\mathbf{C}(t,:).$$

Sampling approach:

1. Fix a probability distribution $\{p_t\}_{t=1}^{T}$ on the index vector $\{1,2,\dots,T\}$.
2. Draw a subset of k indices $J=\{t_1,t_2,\dots,t_k\}\subseteq\{1,2,\dots,T\}$.
3. Use $\bar{\mathbf{A}}=\frac{1}{k}\sum_{i=1}^{k}\frac{1}{p_{t_i}}\mathbf{B}(:\,,t_i)\mathbf{C}(t_i,\,:\,)$ to approximate **A**.

You get an unbiased estimator regardless of the probability distribution. But the computational profile depends critically how which one you choose. Common choices:

Uniform distribution: Very fast. Not very reliable or accurate.

Sample according to column/row norms: Cost is $O(mnk)$, which is much better than $O(mnT)$ when $k \ll T$. Better outcomes than uniform, but not great in general case.

In either case, you need $k \sim \dfrac{1}{\epsilon^2}$ to attain precision ϵ.

Matrix approximation by sampling: Low rank approximation.

Given an $m \times n$ matrix $\mathbf{A}$, we seek a rank-k matrix $\bar{\mathbf{A}}$ such that $\|\mathbf{A} - \bar{\mathbf{A}}\|$ is small.

Sampling approach:

1. Draw vectors J and I holding k samples from the column and row indices, resp.
2. Form matrices $\mathbf{C}$ and $\mathbf{R}$ consisting of the corresponding columns and rows
$$\mathbf{C} = \mathbf{A}(:\,,J), \qquad \text{and} \qquad \mathbf{R} = \mathbf{A}(I,:\,).$$
3. Use as your approximation
$$\begin{array}{ccccc} \bar{\mathbf{A}} & = & \mathbf{C} & \mathbf{U} & \mathbf{R}, \\ m \times n & & m \times k & k \times k & k \times n \end{array}$$
where $\mathbf{U}$ is computed from information in $\mathbf{A}(I,J)$. (It should be an approximation to the optimal choice $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$. For instance, $\mathbf{U} = \mathbf{A}(I,J)^{-1}$.)

The computational profile depends crucially on the probability distribution that is used.

Uniform probabilities: Can be *very* cheap. But in general not reliable.

Probabilities from “leverage scores”: Optimal distributions can be computed using the information in the top left and right singular vectors of $\mathbf{A}$. Then quite strong theorems can be proven on the quality of the approximation. Problem: Computing the probability distribution is as expensive as computing a partial SVD.

Matrix approximation by sampling: Connections to randomized embedding.

Task: Find a rank k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Sampling approach: Draw a subset of k columns $\mathbf{Y} = \mathbf{A}(:, J)$ where J is drawn at random. Let our approximation to the matrix be

$$\mathbf{A}_k = \mathbf{Y}\mathbf{Y}^{\dagger}\mathbf{A}.$$

As we have seen, this in general does not work very well. But it does work well for the class of matrices for which uniform sampling is optimal.

Matrix approximation by sampling: Connections to randomized embedding.

Task: Find a rank k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Sampling approach: Draw a subset of k columns $\mathbf{Y} = \mathbf{A}(:, J)$ where J is drawn at random. Let our approximation to the matrix be

$$\mathbf{A}_k = \mathbf{Y}\mathbf{Y}^{\dagger}\mathbf{A}.$$

As we have seen, this in general does not work very well. But it does work well for the class of matrices for which uniform sampling is optimal. *We can turn* **A** *into such a matrix!*

Matrix approximation by sampling: Connections to randomized embedding.

Task: Find a rank k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Sampling approach: Draw a subset of k columns $\mathbf{Y} = \mathbf{A}(:, J)$ where J is drawn at random. Let our approximation to the matrix be

$$\mathbf{A}_k = \mathbf{Y}\mathbf{Y}^{\dagger}\mathbf{A}.$$

As we have seen, this in general does not work very well. But it does work well for the class of matrices for which uniform sampling is optimal. *We can turn* $\mathbf{A}$ *into such a matrix!* Let $\boldsymbol{\Omega}$ be a matrix drawn from a uniform distribution on the set of $n \times n$ unitary matrices (the "Haar distribution"). Then form

$$\tilde{\mathbf{A}} = \mathbf{A}\boldsymbol{\Omega}.$$

Now each column of $\tilde{\mathbf{A}}$ has exactly the same distribution! We may as well pick $J = 1 : k$, and can then pick a great sample through

$$\mathbf{Y} = \tilde{\mathbf{A}}(:, J) = \mathbf{A}\boldsymbol{\Omega}(:, J).$$

The $n \times k$ "slice" $\boldsymbol{\Omega}(:, J)$ is in a sense an optimal random embedding.

Fact: Using a Gaussian matrix is mathematically equivalent to using $\boldsymbol{\Omega}(:, J)$.

Question: What other choices of random projection might mimic the action of $\boldsymbol{\Omega}(:, J)$?

Matrix approximation by sampling: Structured random embeddings

Task: Find a rank k approximation to a given $m \times n$ matrix $\mathbf{A}$.

Approach: Draw an $n \times k$ random embedding $\mathbf{\Omega}$, set $\mathbf{Y} = \mathbf{A\Omega}$, and then form $\mathbf{A}_k = \mathbf{YY}^{\dagger}\mathbf{A}$.

Choices of random embeddings:

- *Gaussian (or slice of Haar matrix):* Optimal. Leads to $O(mnk)$ overall cost.
- *Subsampled randomized Fourier transform (SRFT):* Indistinguishable from Gaussian in practice. Leads to $O(mn\log(k))$ overall cost. Adversarial counter examples can be built, so supporting theory is weak.
- *Chains of Givens rotations:* Similar profile to an SRFT.
- *Sparse random projections:* Need at least two nonzero entries per row. Works surprisingly well.
- *Additive random projections:* You can use a map with only ±1 entries.

Matrix approximation by sampling: Key points

- These techniques provide a path forwards for problems where traditional techniques are not viable. Examples of applications:

  - Kernel matrices in data analysis. Matrices are dense, and you cannot afford to form the entire matrix. “Kernel ridge regression”.
  - Kronecker product matrices that arise in tensor approximations.
  - Vast linear systems arising in quantum chemistry.

  In all these examples, there is reason to expect the data to be amenable to sampling.

- Popular topic for theory papers.

- When they are viable, techniques based on randomized embeddings are preferable; they yield higher accuracy, and less variability in the outcome.

Outline

- **Methods based on randomized embeddings**
  - Low rank approximation: The randomized SVD.
  - Streaming and single-pass methods.
  - A posteriori error estimation.
  - Structured random maps (“fast Johnson-Lindenstrauss transforms”).
  - Linear solvers: Sketch-to-solve vs. Sketch-to-precondition.
  - Las Vegas style methods: Extremely robust and reliable.

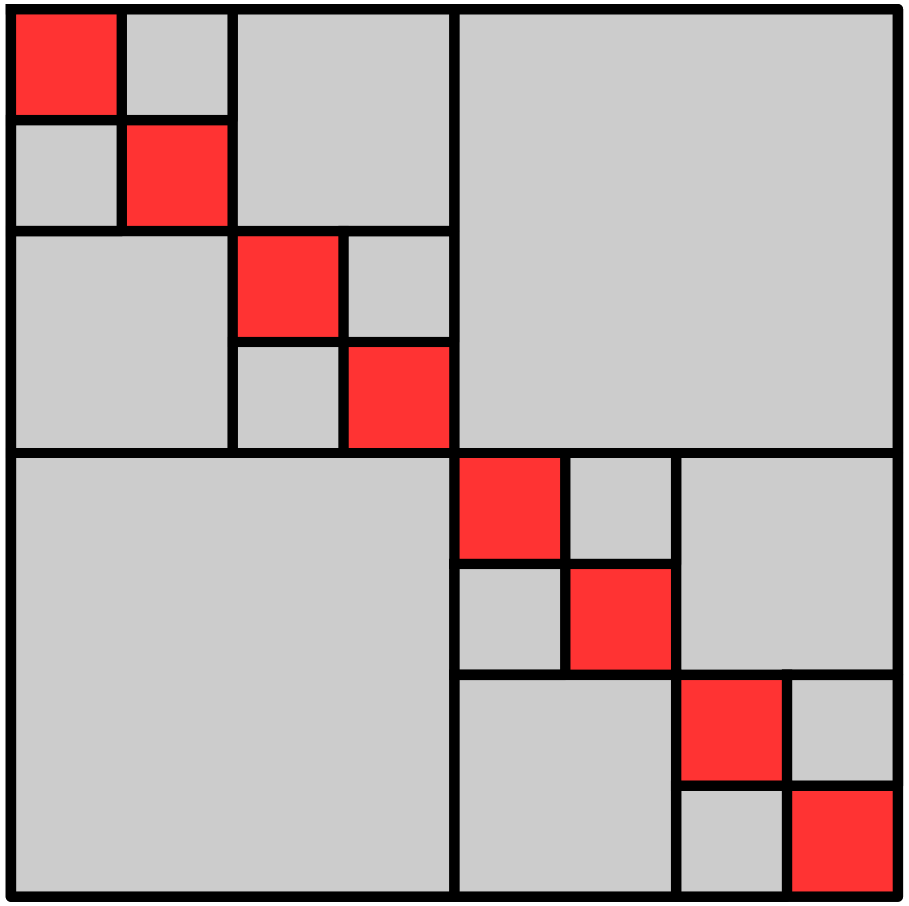
- **Methods based on randomized sampling**
  - Monte Carlo style methods → less reliable, less accurate, less robust.
  - Enable the solution of stupendously large problems that would otherwise be intractable.

- **Approximation of rank-structured matrices**
  - Approximation of global operators of mathematical physics (solution operators, DtNs, . . .).
  - Tools for matrices that are not of global low rank, but have structure that can be exploited.
  - Conceptually related to Fast Multipole Methods, Fast Direct Solvers, Calderón-Zygmund theory . . .

Rank-structured matrices

We use the term *rank-structured* to refer to matrices that are not themselves of globally low rank, but can be tessellated into sub-blocks in such as way that each block is either small or of low numerical rank.

We focus on "hierarchical" tessellations (as the one shown on the right). Some techniques apply to "flat" formats as well.



All gray blocks have low rank.

Hierarchically rank-structured matrices often admit linear or close to linear complexity algorithms for the matrix-vector multiply, matrix-matrix multiply, LU factorization, etc.

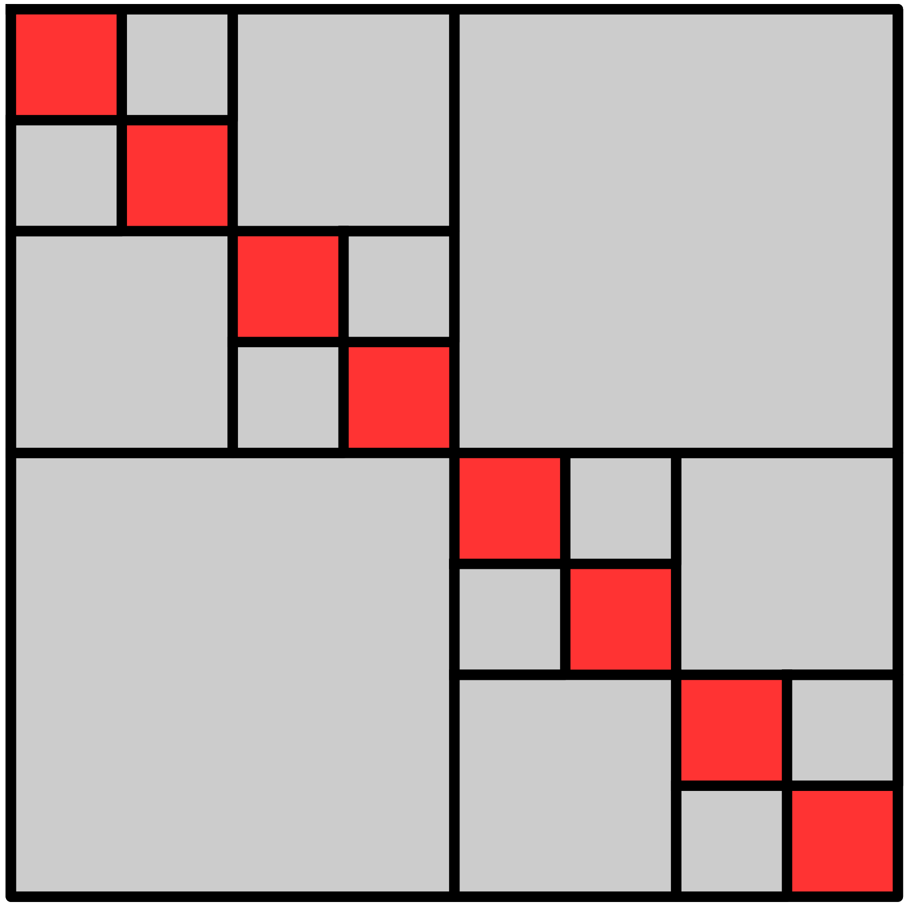
Ubiquitous applications in scientific computing: *Solution operators for elliptic PDEs, DtN operators, scattering matrices, Schur complements in sparse direct solvers, etc.*

More recently, have been shown to arise in data science as well — kernel matrices, covariance matrices, Hessians, etc.

Rank-structured matrices

We use the term *rank-structured* to refer to matrices that are not themselves of globally low rank, but can be tessellated into sub-blocks in such as way that each block is either small or of low numerical rank.

We focus on "hierarchical" tessellations (as the one shown on the right). Some techniques apply to "flat" formats as well.



All gray blocks have low rank.

Hierarchically rank-structured matrices often admit linear or close to linear complexity algorithms for the matrix-vector multiply, matrix-matrix multiply, LU factorization, etc.

Ubiquitous applications in scientific computing: *Solution operators for elliptic PDEs, DtN operators, scattering matrices, Schur complements in sparse direct solvers, etc.*

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, ...); ...

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

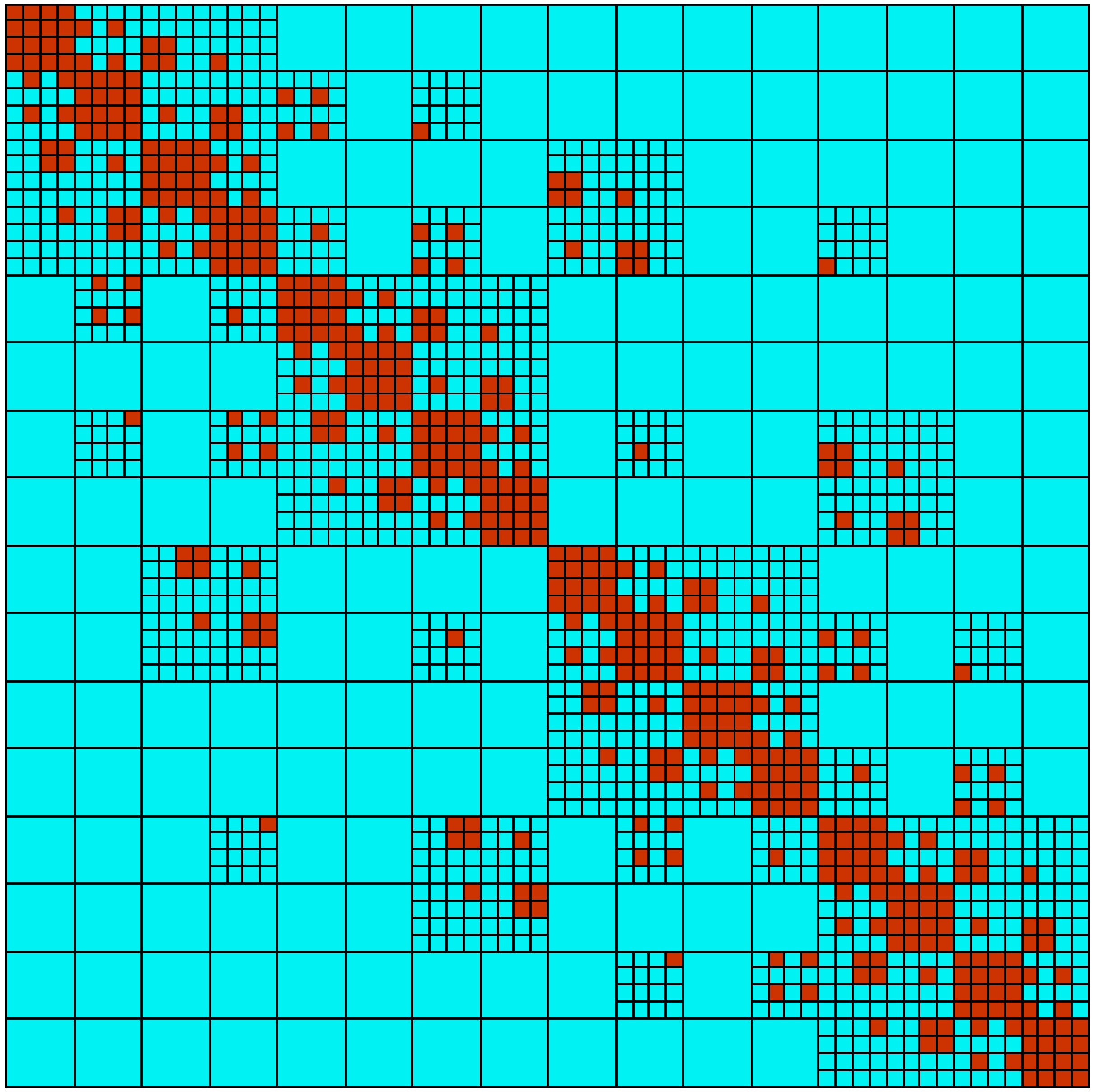


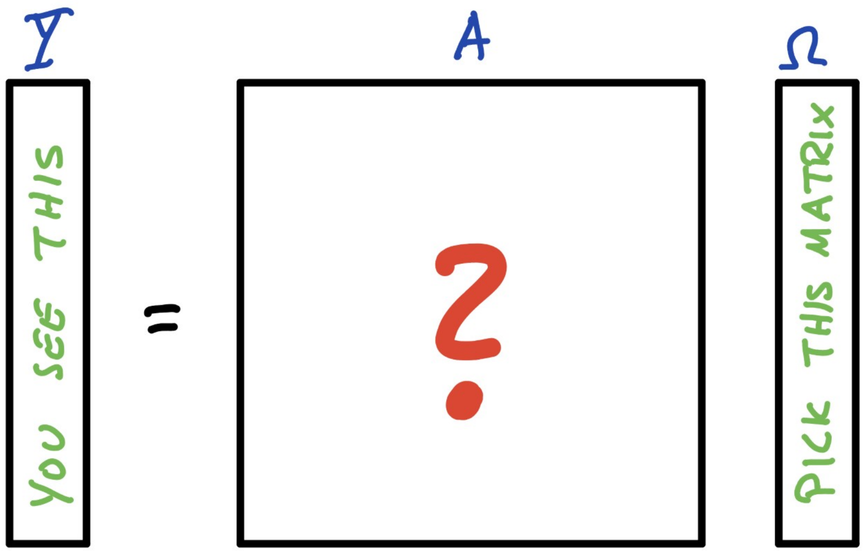
Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

Approximation of rank-structured matrices

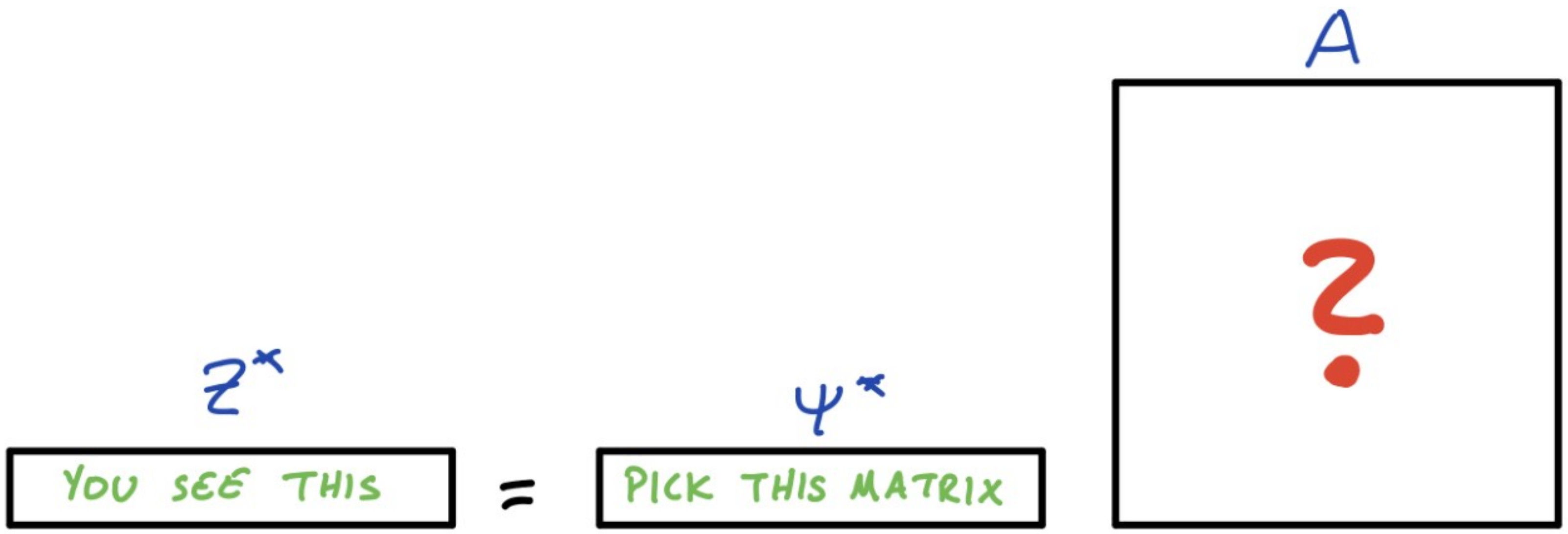
Environment: We are given a rank-structured matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

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:



Approximation of rank-structured matrices

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 low rank case: In the particularly simple case where $\mathbf{A}$ has *global* rank k, we revert to the case we considered in the first part of the talk.

In the current framework, the randomized SVD takes the form:

- Set $s = k$ and draw a "test matrix" $\mathbf{\Omega} \in \mathbb{R}^{N\times s}$ from a Gaussian distribution.
- Form the "sample matrix" $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$.
- Build $\mathbf{\Psi}$ to hold an ON basis for $\mathrm{ran}(\mathbf{Y})$, e.g., $[\mathbf{\Psi}, \sim] = \mathtt{qr}(\mathbf{Y}, 0)$.
- Form $\mathbf{Z} = \mathbf{A}^{*}\mathbf{\Psi}$.

Then $\mathbf{A} = \mathbf{\Psi}\left(\mathbf{\Psi}^{*}\mathbf{A}\right) = \mathbf{\Psi}\mathbf{Z}^{*}$ with probability 1.

In the more typical case where $\mathbf{A}$ is only *approximately* of rank k, some *oversampling* is required to get a reliable scheme. (Say $s = k + 10$, or $s = 2k$, or some such.)

Rank structured case: Extract *all* the low-rank matrices, and *all* the dense blocks, from a very limited set of global "probes". How do you disentangle the mixed samples?

Approximation of rank-structured matrices

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

Why generalize from "global low rank" to "rank-structured":

- Integral operators from classical physics. If you have a legacy method for the matrix-vector multiple (e.g. the Fast Multipole Method), then we could enable a range of operations – LU factorization, matrix inversion, etc.
- Multiplication of operators. Useful for forming Dirichlet-to-Neumann operators, for combining solvers of multi-physics problems, etc.
- Compression of Schur complements that arise in the LU or Cholesky factorization of sparse matrices. This overcomes the key computational bottleneck, and for instance admits the acceleration of the LU factorization of a "finite element" matrix *from $O(N^2)$ to near linear complexity.*

Approximation of rank-structured matrices

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 $\mathbf{Z} = \mathbf{A}^{*}\mathbf{\Psi}$?

Available techniques for the rank structured case:

For the most general structured matrix formats (e.g. $\mathcal{H}$-matrices), the problem has been solved in principle, and close to linear complexity algorithms exist:

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

However, existing methods require $\sim k\,\log(N)$ matvecs, and do not have great practical speed. For instance, as dimension d increases, the bound on flops has an 8^{d} factor …

Recently proposed algorithms have reduced the pre-factors by constructing bespoke random matrices that are designed to be optimal for any given tessellation pattern. The key technical idea is to formulate admissibility criteria that form a graph, and then exploit powerful graph coloring algorithms. This technique also enables compression of kernel matrices that arise in ML. *[J. Levitt & P.G. Martinsson, JCAM **451**(1), 2024.]*

Approximation of rank-structured matrices

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 $\mathbf{Z} = \mathbf{A}^{*}\mathbf{\Psi}$?

Available techniques for the rank structured case:

The good news is that in the context of *numerical PDEs*, more specialized rank structured formats are often sufficient — hierarchically semi-separable matrices, hierarchically block-separable matrices, “$\mathcal{H}$-matrices with weak admissibility”, etc.

For these matrices, algorithms with true linear complexity and high practical speed exist.

First generation algorithms were not fully black box, as they required the ability to evaluate a small number of matrix entries explicitly.

- P.G. Martinsson, SIMAX, **32**(4), 2011.
- Later improvements by Jianlin Xia, Sherry Li, and others. Widely used.

Recent: Fully black box algorithm with true linear complexity and high practical speed:

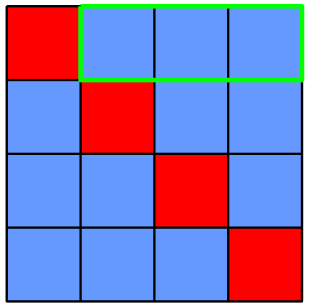
- J. Levitt & P.G. Martinsson, SISC, **46**(3), 2024.

Even more recent: Do inversion and compression simultaneously → order of magnitude reduction in memory requirements. *[Anna Yesypenko PhD thesis, UT-Austin, Nov. 2023]*

Approximation of rank-structured matrices – A naive approach

Consider the task of finding a basis matrix $\mathbf{U}_4$ for node 4 using randomized sampling.

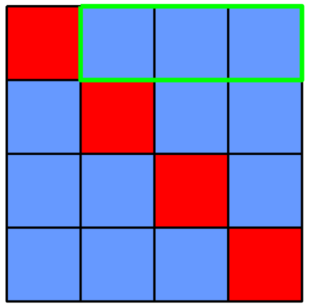
We seek a sample of $\mathbf{A}(I_4, I_4^c)$, the HBS row block of node 4.



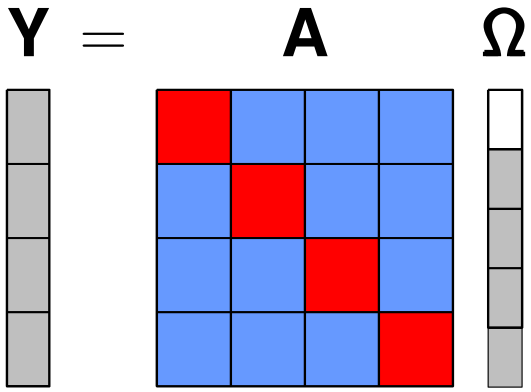
Approximation of rank-structured matrices – A naive approach

Consider the task of finding a basis matrix $\mathbf{U}_4$ for node 4 using randomized sampling.

We seek a sample of $\mathbf{A}(I_4, I_4^c)$, the HBS row block of node 4.



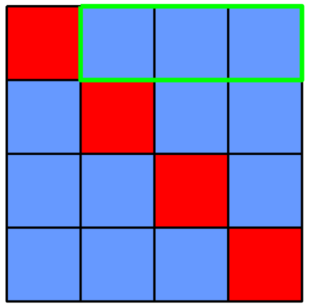
The naive approach is to sample with a random matrix $\mathbf{\Omega} \in \mathbb{R}^{N\times r}$, $r = k + 10$, that has a block of zeros in rows indexed by I_4. Then $\mathbf{Y}(I_4, :)$ will contain a sample of $\mathbf{A}(I_4, I_4^c)$.



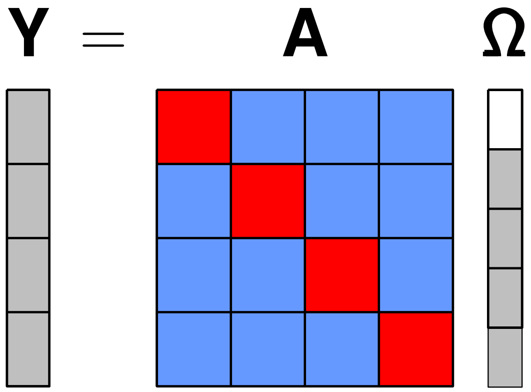
Approximation of rank-structured matrices – A naive approach

Consider the task of finding a basis matrix $\mathbf{U}_4$ for node 4 using randomized sampling.

We seek a sample of $\mathbf{A}(I_4, I_4^c)$, the HBS row block of node 4.



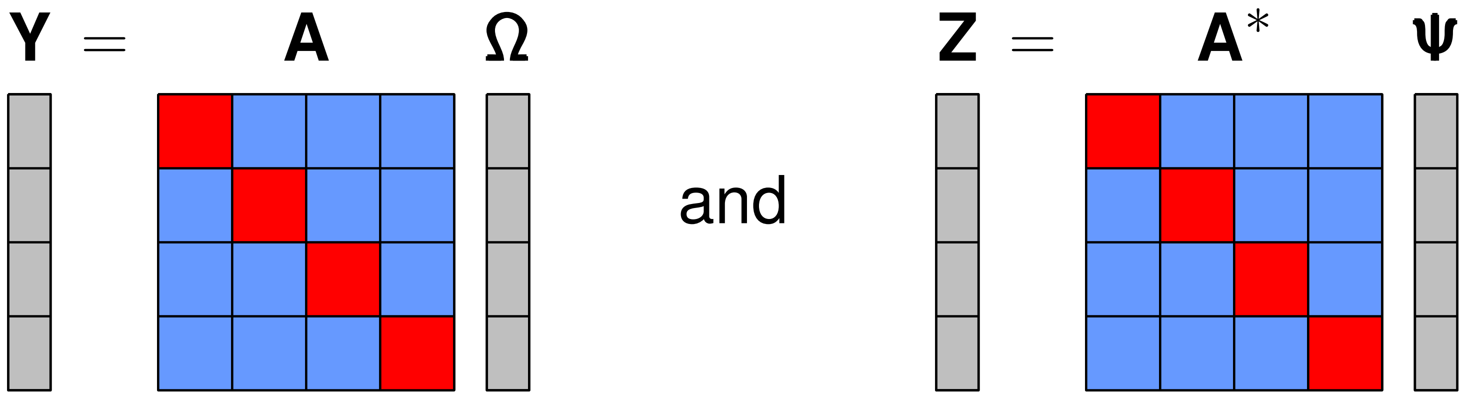
The naive approach is to sample with a random matrix $\mathbf{\Omega} \in \mathbb{R}^{N\times r}$, $r = k + 10$, that has a block of zeros in rows indexed by I_4. Then $\mathbf{Y}(I_4, :)$ will contain a sample of $\mathbf{A}(I_4, I_4^c)$.



This scheme requires taking a separate set of r samples *for each leaf node*, for a total of $\sim rN/m$ samples. There is a lot of wasted information in $\mathbf{Y}$.

Approximation of rank-structured matrices – the “almost” black-box case

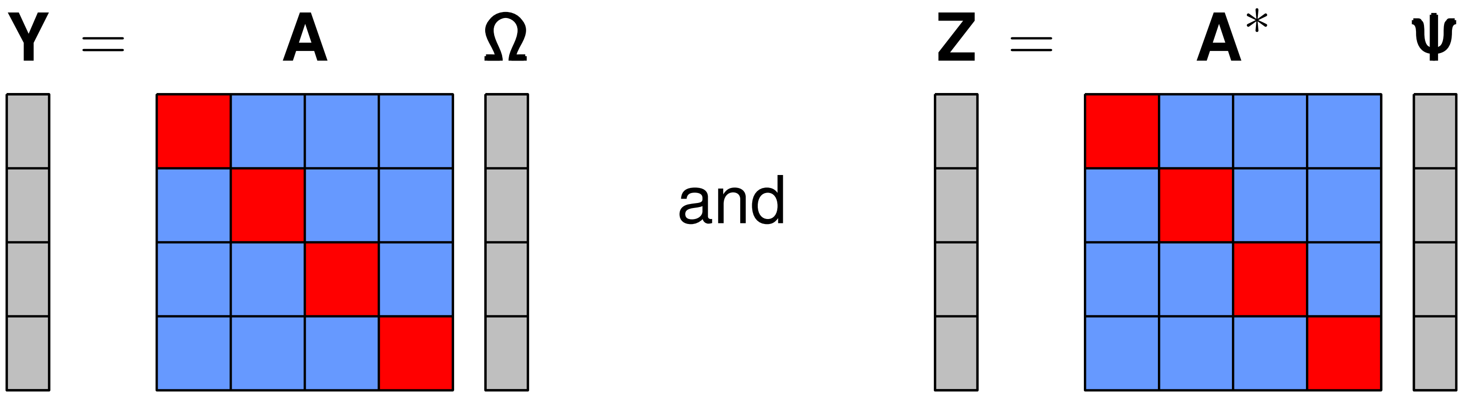
Sample $\mathbf{A}$ with *fixed* dense random matrices $\boldsymbol{\Omega} \in \mathbb{R}^{N\times r}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{N\times r}$:


$\mathbf{Y} = \mathbf{A}\,\boldsymbol{\Omega}$
and
$\mathbf{Z} = \mathbf{A}^{*}\,\boldsymbol{\Psi}$

Assumption: You can do matvecs and *entry evaluation*. (More general soon.)

Approximation of rank-structured matrices – the “almost” black-box case

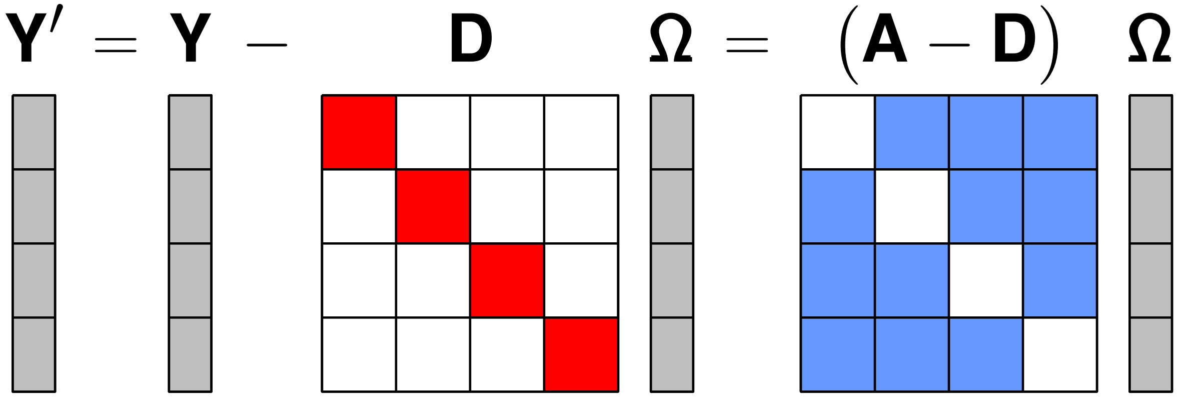
Sample $\mathbf{A}$ with *fixed* dense random matrices $\mathbf{\Omega} \in \mathbb{R}^{N\times r}$ and $\mathbf{\Psi} \in \mathbb{R}^{N\times r}$:



$$\mathbf{Y} = \mathbf{A}\,\mathbf{\Omega} \qquad \text{and} \qquad \mathbf{Z} = \mathbf{A}^{*}\,\mathbf{\Psi}$$

Assumption: You can do matvecs and *entry evaluation*. (More general soon.)

In this case, we can explicitly form the diagonal blocks, and subtract their contributions:



$$\mathbf{Y}' = \mathbf{Y} - \mathbf{D}\,\mathbf{\Omega} = (\mathbf{A} - \mathbf{D})\,\mathbf{\Omega}$$

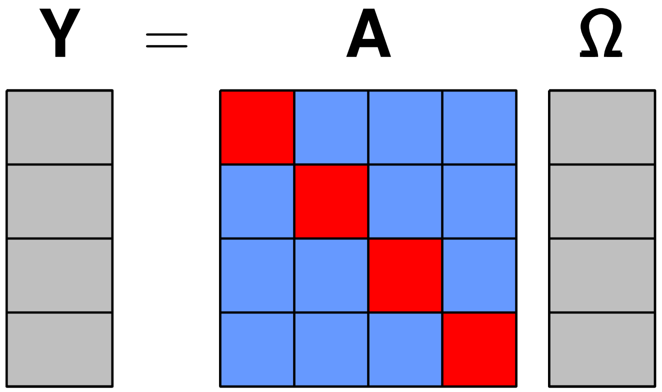
Processing $\mathbf{Z}$ analogously, we obtain basis matrices $\mathbf{Y}_j$ and $\mathbf{Z}_j$ for $j \in \{4, 5, 6, 7\}$ such that

$$\mathbf{A}_{i,j} \approx \mathbf{Y}_i\,\mathbf{B}_{i,j}\,\mathbf{Z}_j^{*}, \qquad i \neq j,$$

for *some* small matrices $\mathbf{B}_{i,j}$. How do you find them?

Approximation of rank-structured matrices – fully black box

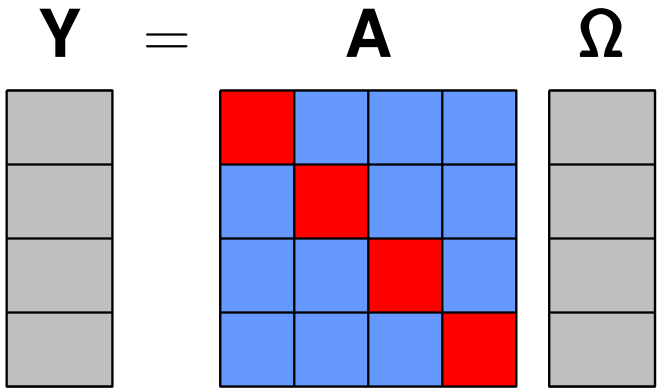
Sample $\mathbf{A}$ with a completely dense random matrix $\mathbf{\Omega} \in \mathbb{R}^{N\times(r+m)}$, where m is the leaf node size. (Think $m \approx 2k$ and $r = k + 10$.)



Let us consider the problem of finding a basis matrix $\mathbf{U}_4$ for the block $\mathbf{A}(I_4, I_4^c)$.

Approximation of rank-structured matrices – fully black box

Sample $\mathbf{A}$ with a completely dense random matrix $\mathbf{\Omega} \in \mathbb{R}^{N\times(r+m)}$, where m is the leaf node size. (Think $m \approx 2k$ and $r = k + 10$.)



Let us consider the problem of finding a basis matrix $\mathbf{U}_4$ for the block $\mathbf{A}(I_4, I_4^c)$.

Since $\mathbf{\Omega}(I_4, :\,)$ is of size $m \times (r+m)$, it has a nullspace of dimension at least r. Let

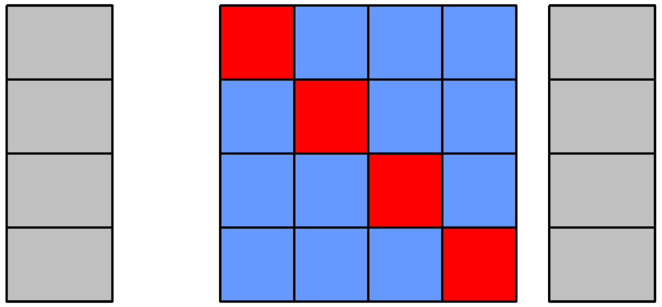
$$\mathbf{Q}_4 = \texttt{nullspace}(\mathbf{\Omega}(I_4, :\,), r)$$

be an $(r+m) \times r$ orthonormal basis of the nullspace of $\mathbf{\Omega}(I_4, :\,)$.

Approximation of rank-structured matrices – fully black box

Sample **A** with a completely dense random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{N\times(r+m)}$, where m is the leaf node size. (Think $m \approx 2k$ and $r = k + 10$.)

$$\mathbf{Y} \quad = \quad \mathbf{A} \quad \boldsymbol{\Omega}$$

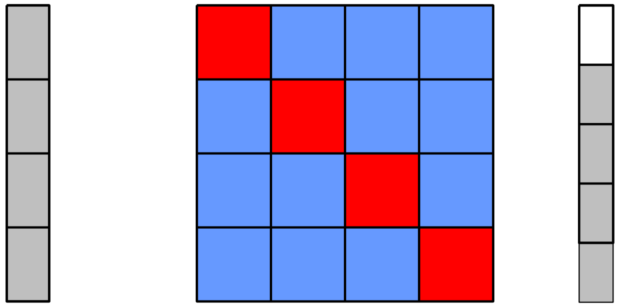


Let us consider the problem of finding a basis matrix $\mathbf{U}_4$ for the block $\mathbf{A}(I_4, I_4^c)$. Since $\boldsymbol{\Omega}(I_4, :\ )$ is of size $m \times (r+m)$, it has a nullspace of dimension at least r. Let

$$\mathbf{Q}_4 = \texttt{nullspace}(\boldsymbol{\Omega}(I_4, :\ ), r)$$

be an $(r+m) \times r$ orthonormal basis of the nullspace of $\boldsymbol{\Omega}(I_4, :\ )$.Then

$$\mathbf{Y}\mathbf{Q}_4 \quad = \quad \mathbf{A} \quad \boldsymbol{\Omega}\mathbf{Q}_4.$$



Orthonormalizing the sample gives basis matrix $\mathbf{U}_4$,

$$\mathbf{U}_4 = \texttt{qr}(\mathbf{Y}(I_4, :)\mathbf{Q}_4).$$

Approximation of rank-structured matrices – fully black box

- For each leaf node τ, we compute

$$\mathbf{Q}_\tau = \texttt{nullspace}(\mathbf{\Omega}(I_\tau,:), r)$$

$$\mathbf{U}_\tau = \texttt{qr}(\mathbf{Y}(I_\tau,:)\mathbf{Q}_\tau).$$

- $\mathbf{U}_\tau$ only depends on $\mathbf{\Omega}(I_\tau,:)$ and $\mathbf{Y}(I_\tau,:)$.
- We only need $r+m$ samples to find $\mathbf{U}_\tau$ for every leaf node τ.
- $\mathbf{\Omega}\mathbf{Q}_\tau$ is a Gaussian random matrix, except for the block intentionally zeroed out.

Approximation of rank-structured matrices – fully black box

Recall the telescoping factorization $\mathbf{A} = \mathbf{U}^{(L)}\tilde{\mathbf{A}}^{(L)}(\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)}$.

Steps:

1. Find $\mathbf{U}^{(L)}, \mathbf{V}^{(L)}$.
2. Find $\mathbf{D}^{(L)}$.
3. Compress $\tilde{\mathbf{A}}^{(L)}$ recursively.

```
Compute randomized samples of A and A*.
Form Gaussian random random matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$ of size $N \times s$.
Multiply $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$.

Compress level by level from finest to coarsest.
for level $\ell = L, L-1, \dots, 0$ do
    for node $\tau$ in level $\ell$ do
        if $\tau$ is a leaf node then
            $\mathbf{\Omega}_\tau = \mathbf{\Omega}(I_\tau, :), \quad \mathbf{\Psi}_\tau = \mathbf{\Psi}(I_\tau, :)$
            $\mathbf{Y}_\tau = \mathbf{Y}(I_\tau, :), \quad \mathbf{Z}_\tau = \mathbf{Z}(I_\tau, :)$
        else
            Let $\alpha$ and $\beta$ denote the children of $\tau$.
            $\mathbf{\Omega}_\tau = \begin{bmatrix} \mathbf{V}_\alpha^*\mathbf{\Omega}_\alpha \\ \mathbf{V}_\beta^*\mathbf{\Omega}_\beta \end{bmatrix}, \quad \mathbf{\Psi}_\tau = \begin{bmatrix} \mathbf{U}_\alpha^*\mathbf{\Psi}_\alpha \\ \mathbf{U}_\beta^*\mathbf{\Psi}_\beta \end{bmatrix}$
            $\mathbf{Y}_\tau = \begin{bmatrix} \mathbf{U}_\alpha^*(\mathbf{Y}_\alpha - \mathbf{D}_\alpha\mathbf{\Omega}_\alpha) \\ \mathbf{U}_\beta^*(\mathbf{Y}_\beta - \mathbf{D}_\beta\mathbf{\Omega}_\beta) \end{bmatrix}, \quad \mathbf{Z}_\tau = \begin{bmatrix} \mathbf{V}_\alpha^*(\mathbf{Z}_\alpha - \mathbf{D}_\alpha^*\mathbf{\Psi}_\alpha) \\ \mathbf{V}_\beta^*(\mathbf{Z}_\beta - \mathbf{D}_\beta^*\mathbf{\Psi}_\beta) \end{bmatrix}$

        if level $\ell > 0$ then
            $\mathbf{Q}_\tau = \texttt{nullspace}(\mathbf{\Omega}_\tau, r), \quad \mathbf{P}_\tau = \texttt{nullspace}(\mathbf{\Psi}_\tau, r)$
            $\mathbf{U}_\tau = \texttt{qr}(\mathbf{Y}_\tau\mathbf{Q}_\tau, r), \quad \mathbf{V}_\tau = \texttt{qr}(\mathbf{Z}_\tau\mathbf{P}_\tau, r)$
            $\mathbf{D}_\tau = (\mathbf{I} - \mathbf{U}_\tau\mathbf{U}_\tau^*)\mathbf{Y}_\tau\mathbf{\Omega}_\tau^\dagger + \mathbf{U}_\tau\mathbf{U}_\tau^*\left((\mathbf{I} - \mathbf{V}_\tau\mathbf{V}_\tau^*)\mathbf{Z}_\tau\mathbf{\Psi}_\tau^\dagger\right)^*$
        else
            $\mathbf{D}_\tau = \mathbf{Y}_\tau\mathbf{\Omega}_\tau^\dagger$
```

Approximation of rank-structured matrices – Finding D

From the telescoping factorization

$$\mathbf{A} = \mathbf{U}^{(L)}\tilde{\mathbf{A}}^{(L)}(\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)}$$

we define $\tilde{\mathbf{A}}^{(L)}$ and $\mathbf{D}^{(L)}$ as follows.

Approximation of rank-structured matrices – Finding D

From the telescoping factorization

$$\mathbf{A} = \mathbf{U}^{(L)}\tilde{\mathbf{A}}^{(L)}(\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)}$$

we define $\tilde{\mathbf{A}}^{(L)}$ and $\mathbf{D}^{(L)}$ as follows.

$$\mathbf{A} = \mathbf{U}^{(L)}\overbrace{(\mathbf{U}^{(L)})^*\mathbf{A}\mathbf{V}^{(L)}}^{\tilde{\mathbf{A}}^{(L)}}(\mathbf{V}^{(L)})^* + \overbrace{\mathbf{A} - \mathbf{U}^{(L)}(\mathbf{U}^{(L)})^*\mathbf{A}\mathbf{V}^{(L)}(\mathbf{V}^{(L)})^*}^{\mathbf{D}^{(L)}}$$

Approximation of rank-structured matrices – Finding D

From the telescoping factorization

$$\mathbf{A} = \mathbf{U}^{(L)} \tilde{\mathbf{A}}^{(L)} (\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)}$$

we define $\tilde{\mathbf{A}}^{(L)}$ and $\mathbf{D}^{(L)}$ as follows.

$$\mathbf{A} = \mathbf{U}^{(L)} \overbrace{(\mathbf{U}^{(L)})^* \mathbf{A} \mathbf{V}^{(L)}}^{\tilde{\mathbf{A}}^{(L)}} (\mathbf{V}^{(L)})^* + \overbrace{\mathbf{A} - \mathbf{U}^{(L)} (\mathbf{U}^{(L)})^* \mathbf{A} \mathbf{V}^{(L)} (\mathbf{V}^{(L)})^*}^{\mathbf{D}^{(L)}}$$

Block $\mathbf{D}_\tau$ of $\mathbf{D}^{(L)}$ is given by

$$\begin{aligned} \mathbf{D}_\tau &= \mathbf{A}_{\tau,\tau} - \mathbf{U}_\tau \mathbf{U}_\tau^* \mathbf{A}_{\tau,\tau} \mathbf{V}_\tau \mathbf{V}_\tau^* \\ &= \dots \\ &= (\mathbf{I} - \mathbf{U}_\tau \mathbf{U}_\tau^*) \mathbf{Y}_\tau \mathbf{\Omega}_\tau^\dagger + \mathbf{U}_\tau \mathbf{U}_\tau^* \left((\mathbf{I} - \mathbf{V}_\tau \mathbf{V}_\tau^*) \mathbf{Z}_\tau \mathbf{\Psi}_\tau^\dagger \right)^* \end{aligned}$$

Approximation of rank-structured matrices – Compressing $\tilde{\mathbf{A}}^{(L)}$

To compute randomized samples of $\tilde{\mathbf{A}}^{(L)}$, we multiply the telescoping factorization with $\mathbf{\Omega}$ to obtain

$$\mathbf{Y} = \mathbf{A}\mathbf{\Omega} = (\mathbf{U}^{(L)}\tilde{\mathbf{A}}^{(L)}(\mathbf{V}^{(L)})^* + \mathbf{D}^{(L)})\mathbf{\Omega},$$

and rearrange to obtain

$$\underbrace{(\mathbf{U}^{(L)})^*(\mathbf{Y} - \mathbf{D}^{(L)}\mathbf{\Omega})}_{\text{sample matrix}} = \tilde{\mathbf{A}}^{(L)} \underbrace{(\mathbf{V}^{(L)})^*\mathbf{\Omega}}_{\text{test matrix}}.$$

Key points

- Randomized algorithms for low rank approximation are highly efficient.
  - Interaction with target matrix only through matrix-matrix multiplication → very high practical speed.
  - Particularly efficient for GPUs, out-of-core computing, distributed memory, etc.
  - Structured random maps ("sparse J-L transform"):

    Rank-k approximation in complexity$O(mn\log k)$ or even less (vs. $O(mnk)$ for deterministic).
  - Single pass algorithms have been developed for *streaming environments.*

    Not possible with deterministic methods!

- Randomized algorithms for solving linear systems
  - Overdetermined least squares is particularly successful.
  - The talk only scratched the surface – randomized Kaczmarz, acceleration of Krylov, ...

- Randomized algorithms based on *sampling* make (some) huge problems tractable.
  - Success stories: kernel ridge regression, computational chemistry, tensor approximation, ...

- Randomized compression of rank-structured matrices.
  - Black box randomized algorithms for compressing rank-structured matrices have been established.
  - The combination of "fully black box" and "true linear complexity" was realized only recently.
  - Powerful tools in the construction of fast direct solvers for elliptic PDEs.

Surveys:

- 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.

Software:

- ID: `http://tygert.com/software.html` (ID, SRFT, CPQR, etc)
- RSVDPACK: `https://github.com/sergeyvoronin` (RSVD, randomized ID and CUR)
- HQRRP: `https://github.com/flame/hqrrp/` (LAPACK compatible randomized CPQR)
- Randomized UTV: `https://github.com/flame/randutv`

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