Randomized algorithms for accelerating matrix computations

Per-Gunnar Martinsson

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

University of Texas at Austin

Students, postdocs, collaborators: Chao Chen, Yijun Dong, Abinand Gopal, Nathan Heavner, James Levitt, Yuji Nakatsukasa, Gregorio Quintana-Ortí, Kate Pearce, Joel Tropp, Sergey Voronin, Heather Wilber, Anna Yesypenko.

☞ **Slides:** `http://users.oden.utexas.edu/~pgm/main_talks.html`

Outline:

- **Part 1: Randomized algorithms for low rank approximation** *(survey, 10 min)*
  Randomized singular value decomposition (RSVD) and randomized embeddings.
  Fast Johnson-Lindenstrauss transforms.
  Streaming and out-of-core algorithms.

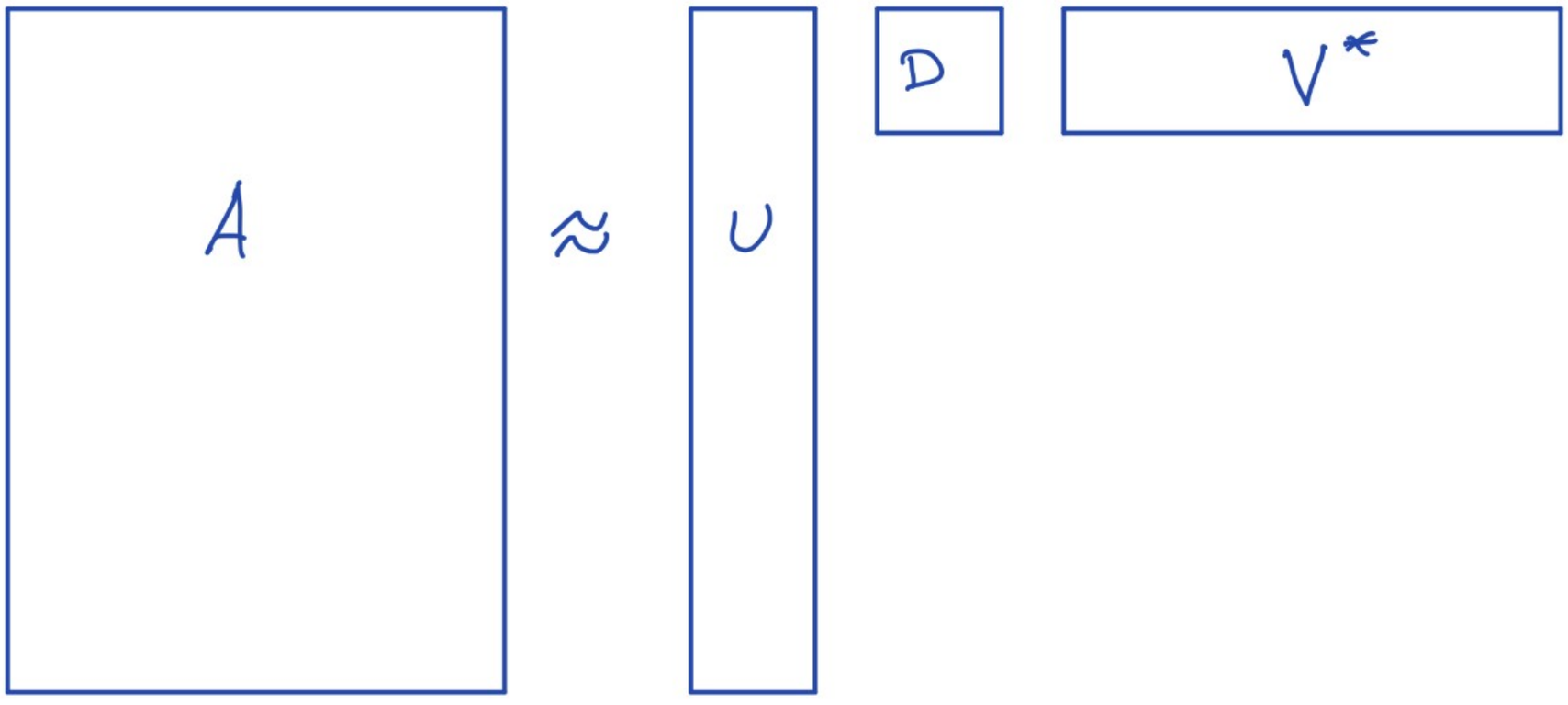
- **Part 2: Compression of rank structured matrices** *(current work, "'blurb", 5? min)*
  Data sparse representations of "kernel matrices".
  Applications in scientific computing and data science.
  Enables $O(N)$ algorithms for many tasks involving large dense matrices.

Randomized algorithms for low rank approximation:

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

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

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



Randomized algorithms for low rank approximation:

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

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

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

Applications:

- Principal component analysis (fitting a hyperplane to data).
- Model reduction in analyzing physical systems.
- PageRank and other spectral methods in data analysis.
- Fast algorithms in scientific computing. (FMMs, Fast Direct Solvers, etc.)
- Diffusion geometry and manifold learning.
- Many, many more ...

Classical deterministic methods: Gram-Schmidt (column pivoted QR), power/subspace iteration, Krylov techniques, ...

Randomized algorithms for low rank approximation:

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

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

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

Solution:

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

Why does it work? When $\mathbf{A}$ has exact rank k, the algorithm succeeds with probability 1.

Randomized algorithms for low rank approximation:

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

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

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

Solution:

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

Why does it work? When $\mathbf{A}$ has exact rank k, the algorithm succeeds with probability 1. In the general case, it fails only if the columns of $\mathbf{G}$ manage to all be close to orthogonal to a dominant right singular vector. *Very* unlikely.

Randomized algorithms for low rank approximation:

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

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

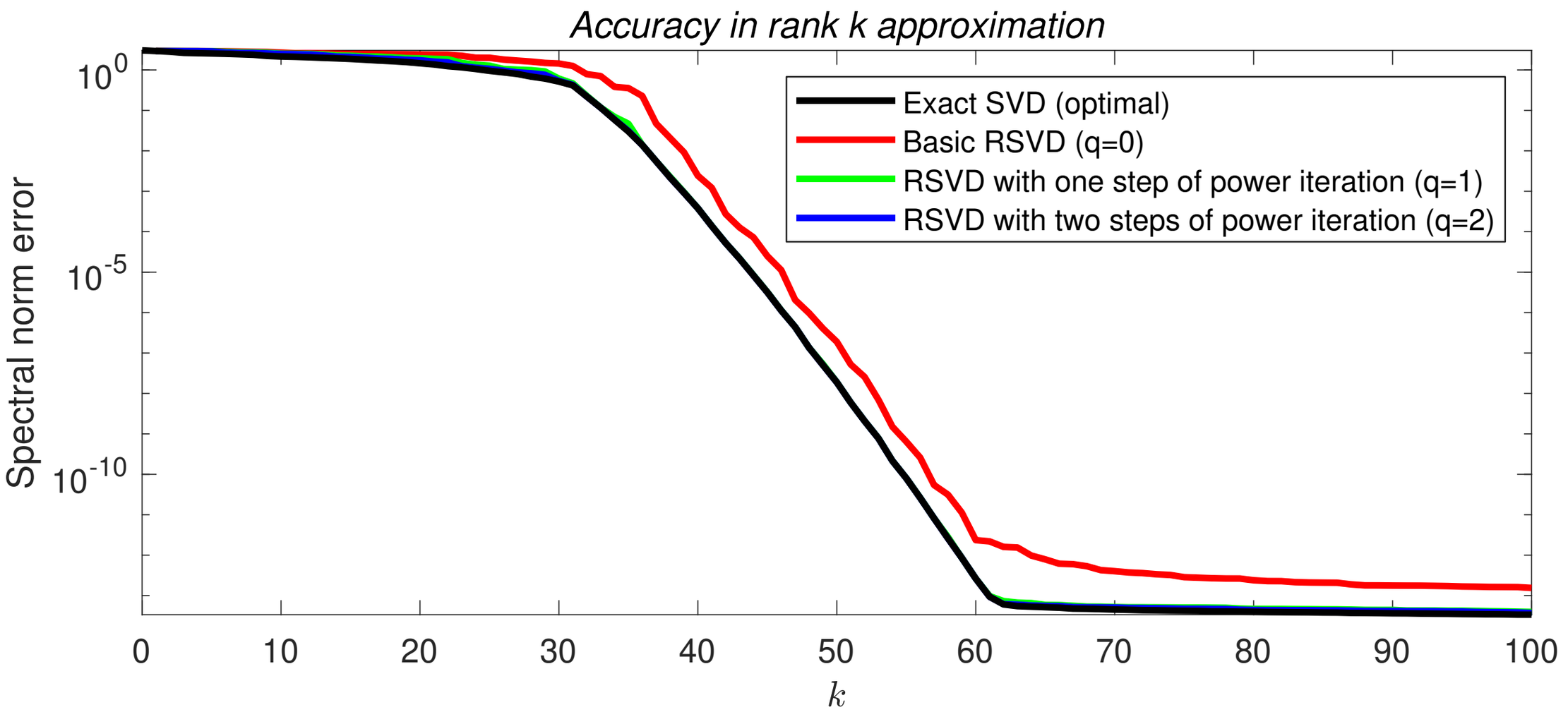
with $\mathbf{U}$ and $\mathbf{V}$ having orthonormal columns, and $\mathbf{D}$ diagonal.

Solution:

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

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

Randomized low rank approximation:



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

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

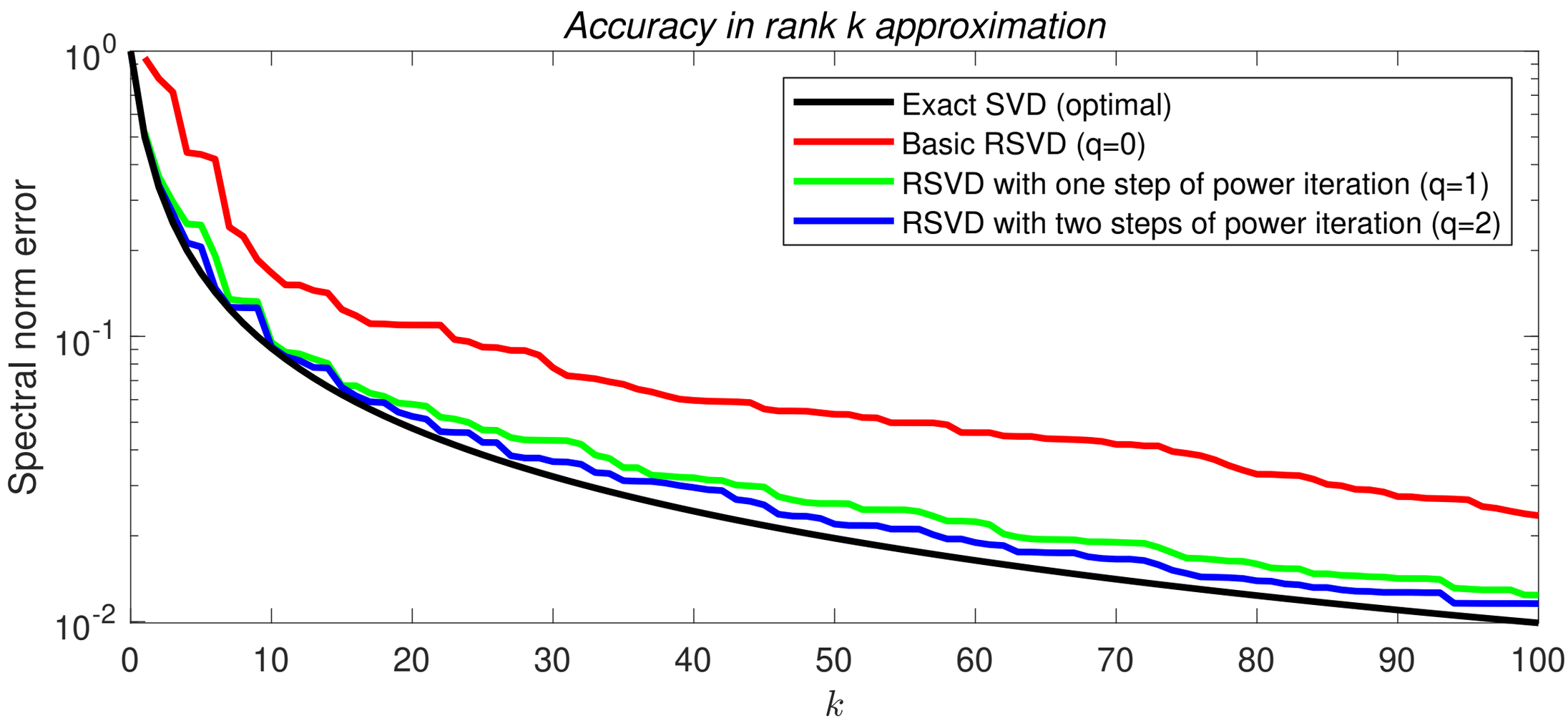
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

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

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

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

Randomized low rank approximation:



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

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

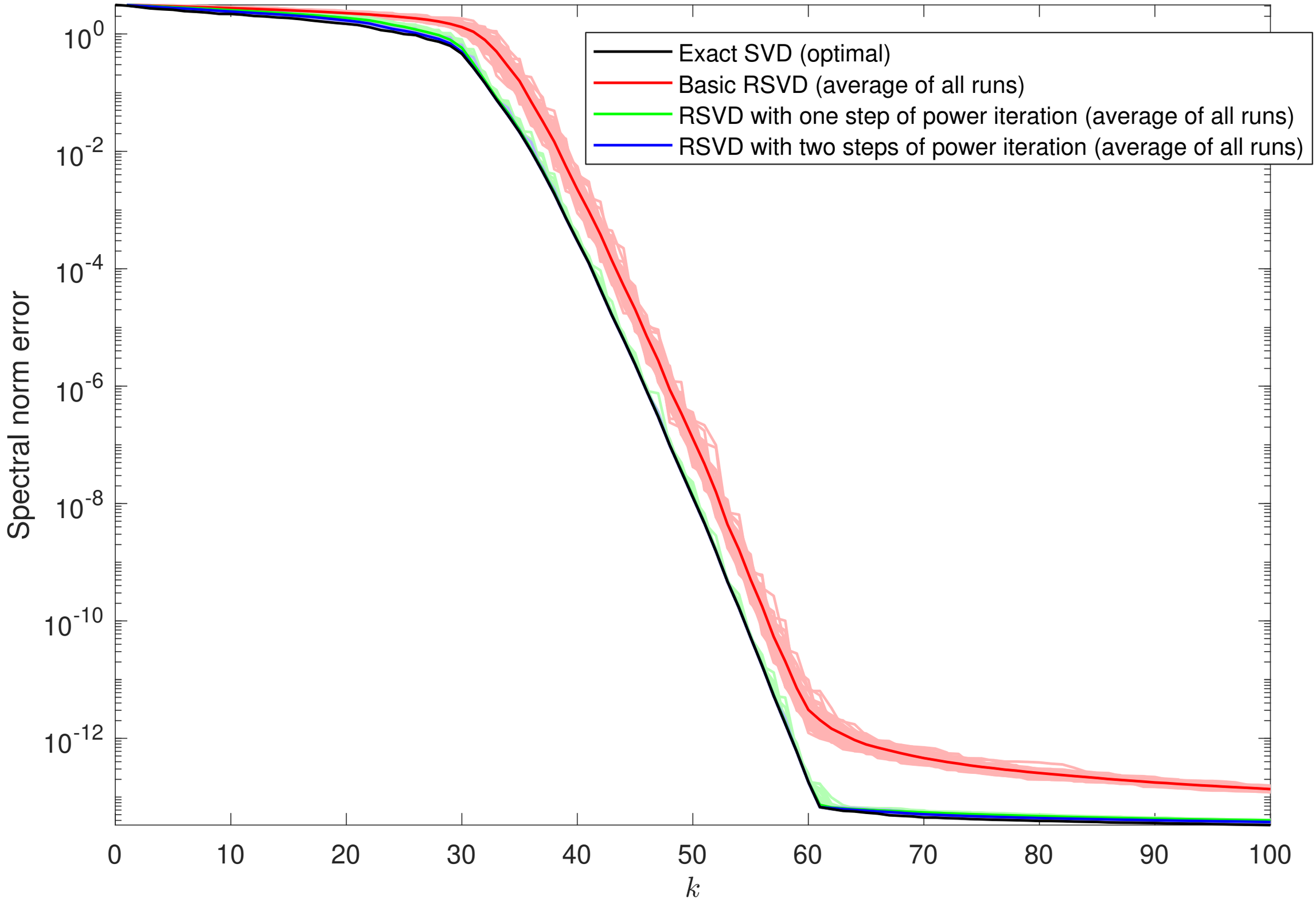
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

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

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

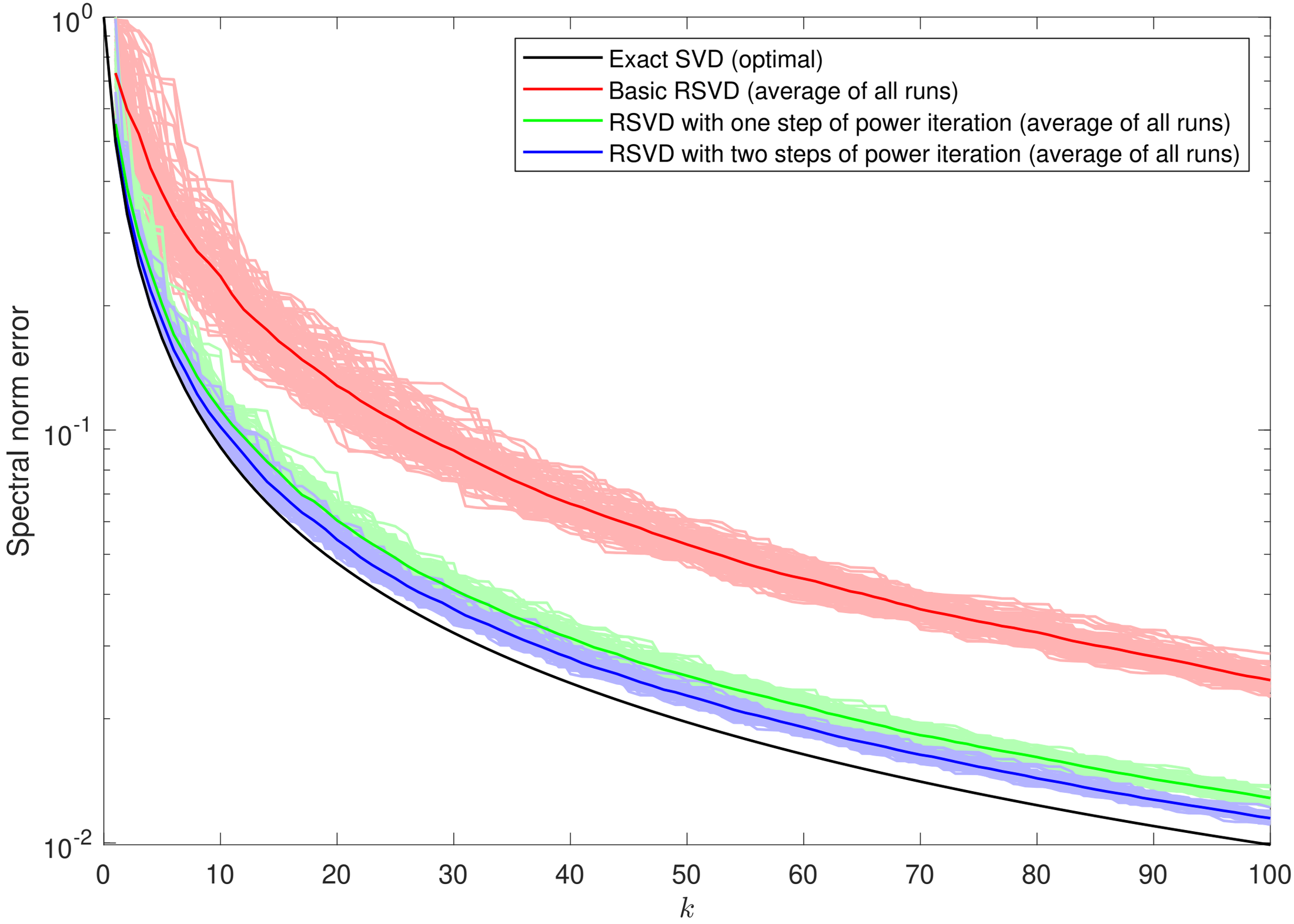
The matrix $\mathbf{A}$ now has singular values that decay slowly.

Randomized low rank approximation: The same plot, but showing 100 instantiations.



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

Randomized low rank approximation: The same plot, but showing 100 instantiations.



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

Input: An $m \times n$ matrix $\mathbf{A}$, 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 $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Oversampling: By drawing a small number p of extra samples, we can prove that the error is close to theoretically minimal. Think $p = 5$ or $p = 10$.

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 $\mathbf{A} \approx \mathbf{UDV}^*$.

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

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

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

(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 $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Oversampling: By drawing a small number p of extra samples, we can prove that the error is close to theoretically minimal. Think $p = 5$ or $p = 10$. For instance, we have:

Theorem: [Halko, Martinsson, Tropp, 2009 & 2011] Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$. Let k denote a target rank and let p denote an over-sampling parameter. Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix. Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AG})$. If $p \geq 2$, then

$$\mathbb{E}\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\text{Frob}} \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2},$$

and

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

There are also bounds on the error when power iteration is used, the likelihood of large deviations from the expected value, and so on.

Focus of current work is construction of *à posteriori* error bounds, and estimates on the accuracy of computed singular *vectors*. (With Yijun Dong and Yuji Nakatsukasa.)

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 $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.

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 $\mathbf{A} \approx \mathbf{UDV}^*$.	
(1) Draw an $n \times (k+p)$ random matrix $\mathbf{G}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix $\mathbf{Q}$ s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

- High practical speed — interacts with $\mathbf{A}$ only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).

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

Key results on randomized SVD:

- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn\log k)$.

  The key is to use a *Fast Johnson-Lindenstrauss transform*.

  Practical acceleration is achieved at ordinary matrix sizes.

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

Key results on randomized SVD:

- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn \log k)$.
- Single pass algorithms have been developed for *streaming environments.*

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

  You cannot store the matrix.

  Not possible with deterministic methods!

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

(1) Draw an $n \times (k+p)$ random matrix **G**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AG}$.	(5) Factor the small matrix $\mathbf{B} = \hat{\mathbf{U}}\mathbf{DV}^*$.
(3) Compute an ON matrix **Q** s.t. $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.	(6) Form $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Key results on randomized SVD:

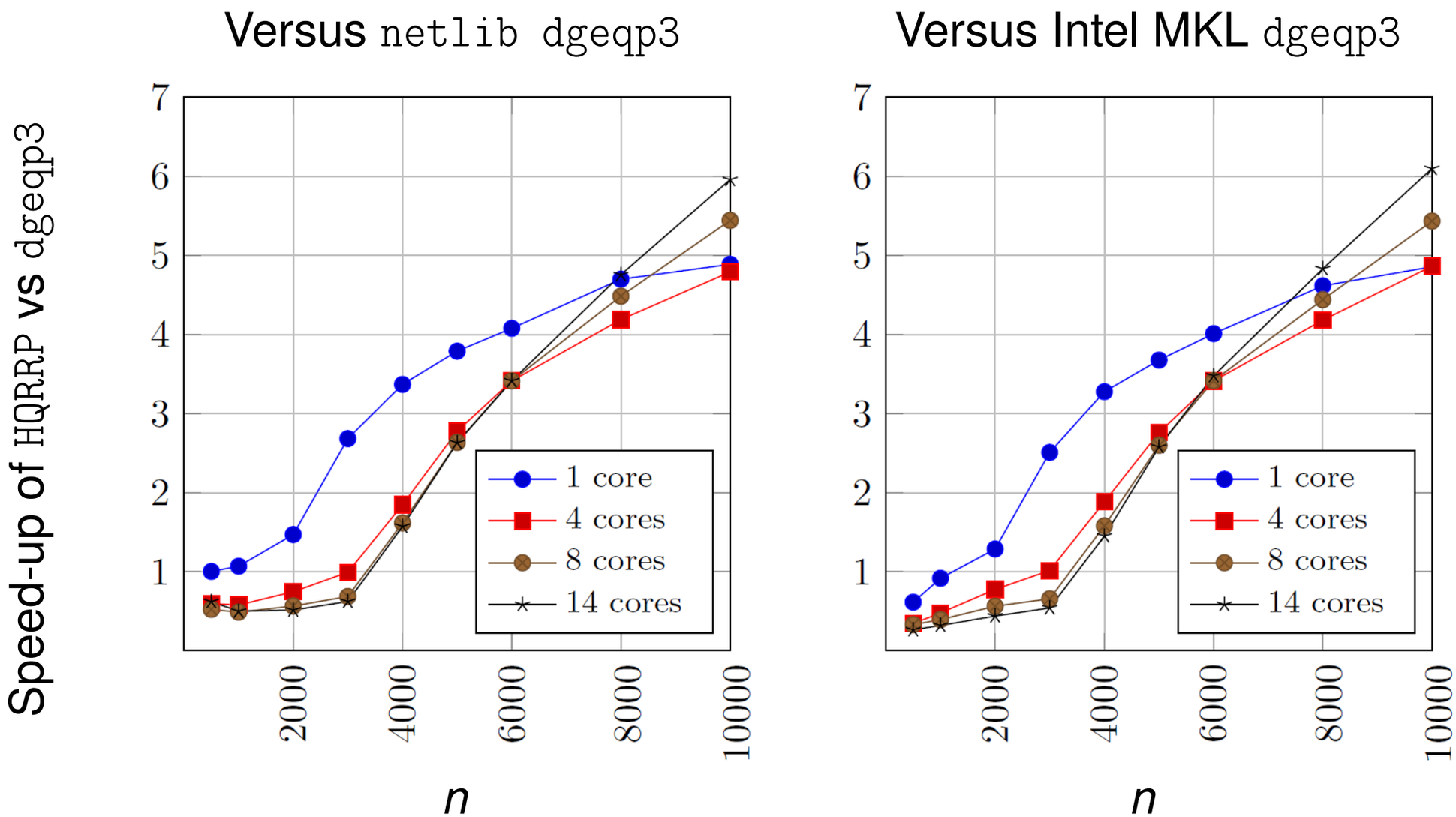
- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Order of magnitude acceleration for data stored *out-of-core.*
- Highly efficient for GPU computing, or mobile computing (phones, etc).
- Consider the problem of computing the dominant k eigenvectors/eigenvalues of a dense matrix of size $m \times n$. Reduction in complexity from $O(mnk)$ to $O(mn \log k)$.
- Single pass algorithms have been developed for *streaming environments.*

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

  You cannot store the matrix.

  Not possible with deterministic methods!
- The randomization idea can be used to overcome a classical problem in numerical linear algebra: How to efficiently implement column-pivoted QR factorizations. (How to cast the computation as BLAS3 operations instead of BLAS2.)

A very fast *randomized* implementation of column pivoted QR



Speedup attained by a randomized algorithm for computing a full column pivoted QR factorization of an $n \times n$ matrix. The speed-up is measured versus LAPACK's faster routine `dgeqp3` *as implemented in Netlib (left) and Intel's MKL (right). Our implementation was done in C, and was executed on an Intel Xeon E5-2695. Joint work with G. Quintana-Ortí, N. Heavner, and R. van de Geijn (SISC 2017). Closely related work by Duersch and Gu, SISC 2017 / SIREV 2020.*

References on randomized SVD:

- N. Halko, P. G. Martinsson, J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, SIAM Review, 2011.
- P.G. Martinsson & J.A. Tropp, *Randomized Numerical Linear Algebra: Foundations & Algorithms*. 2020 Acta Numerica. arxiv #2002.01387.
- P.G. Martinsson, V. Rokhlin and M. Tygert (2006a), A randomized algorithm for the approximation of matrices, Technical Report Yale CS research report YALEU/DCS/RR-1361, Yale.
- Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert, *Randomized algorithms for the low-rank approximation of matrices*. PNAS 2007 104: 20167-20172.
- F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, **25**(3), 2008.
- V. Rokhlin, A. Szlam, and M. Tygert, *A Randomized Algorithm for Principal Component Analysis*, SIAM J. Matrix Anal. Appl., 31(3), 1100–1124, 2009.
- P.G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the approximation of matrices*. Applied and Computational Harmonic Analysis, 30(1), pp. 47–68, 2011.

Relevant prior work in:

- C. H. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala (2000), *Latent semantic indexing: a probabilistic analysis*, Vol. 61, pp. 217–235.
- A. Frieze, R. Kannan and S. Vempala (2004), *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. ACM 51(6), 1025–1041.

Outline:

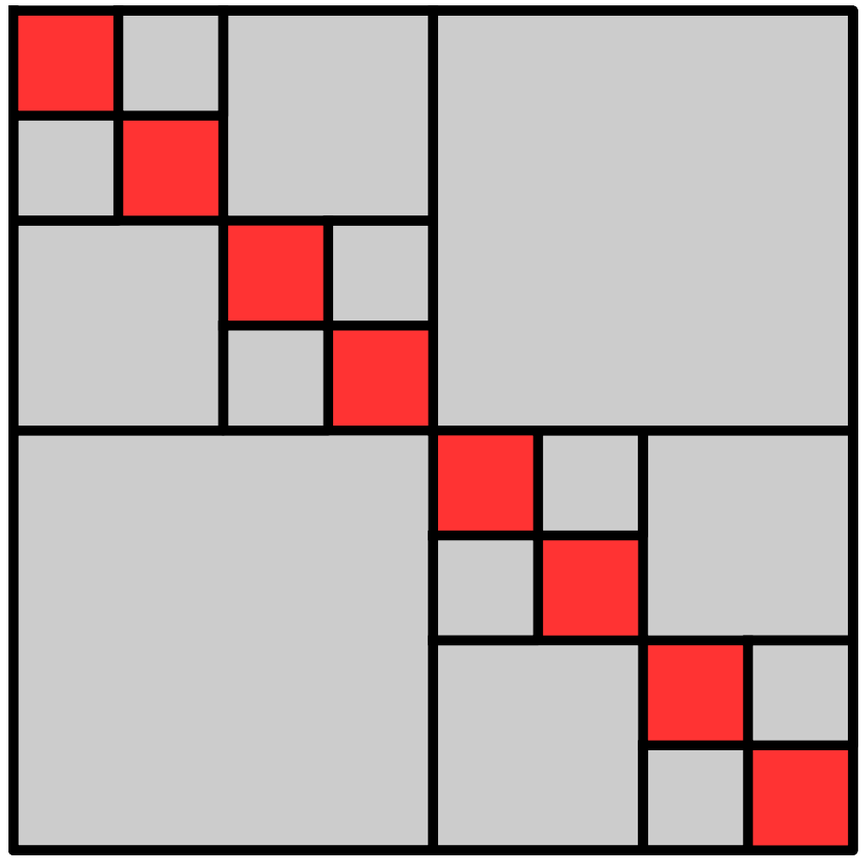
- **Part 1: Randomized algorithms for low rank approximation** *(survey, 10 min)*
  Randomized singular value decomposition (RSVD) and randomized embeddings.
  Fast Johnson-Lindenstrauss transforms.
  Streaming and out-of-core algorithms.

- **Part 2: Compression of rank structured matrices** *(current work, "'blurb", 5? min)*
  Data sparse representations of "kernel matrices".
  Applications in scientific computing and data science.
  Enables $O(N)$ algorithms for many tasks involving large dense matrices.

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.

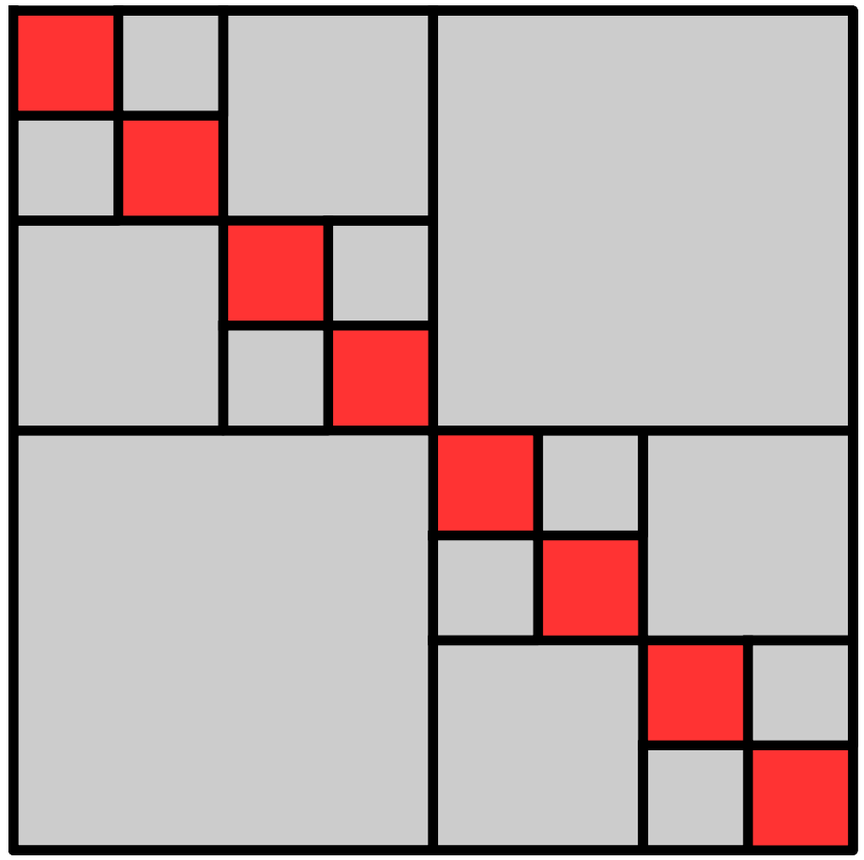
Ubiquitous applications in scientific computing: *Solution operators for elliptic PDEs, DtN operators, scattering matrices, Schur complements in sparse direct solvers, etc.*

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 . . .

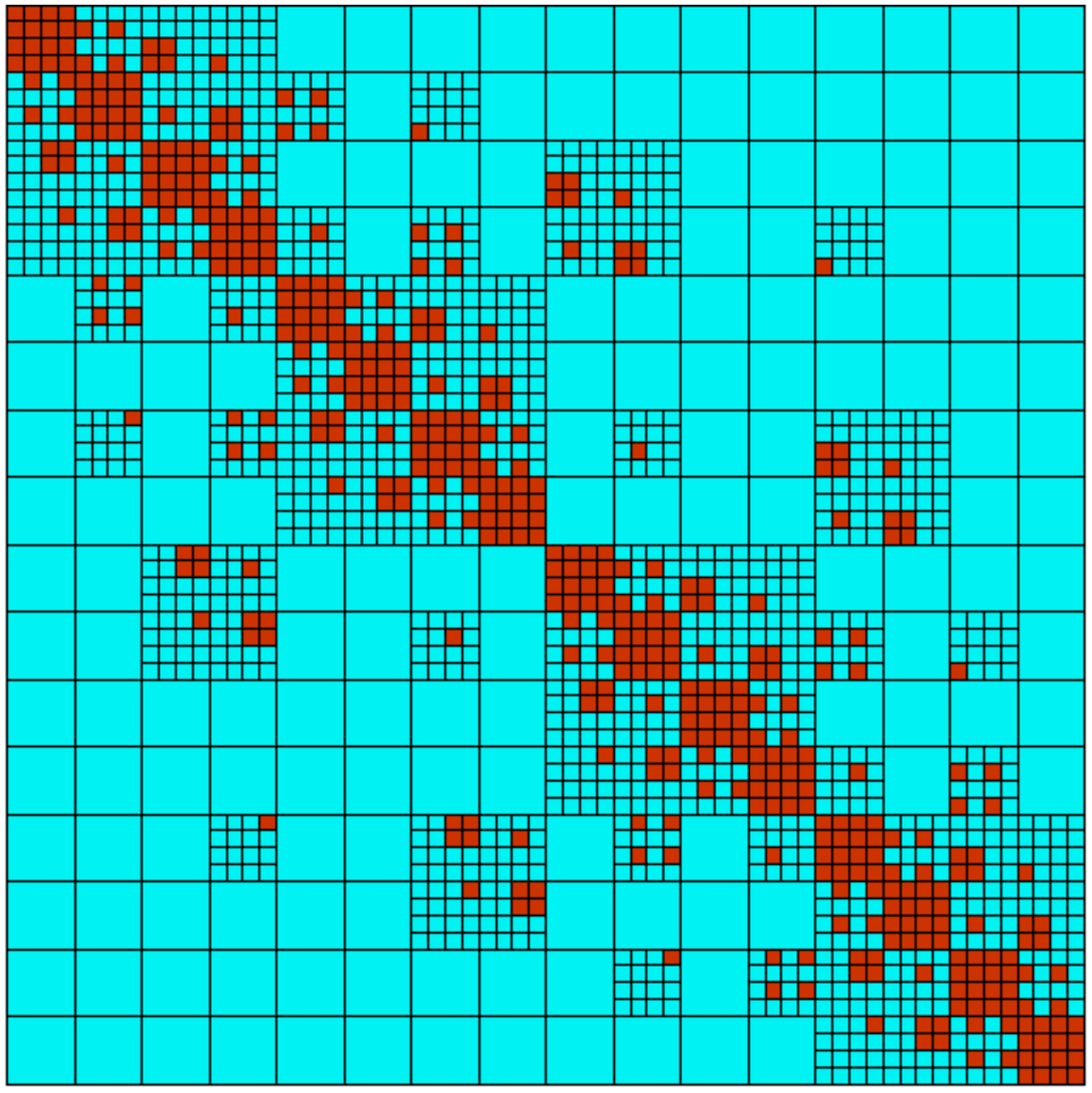


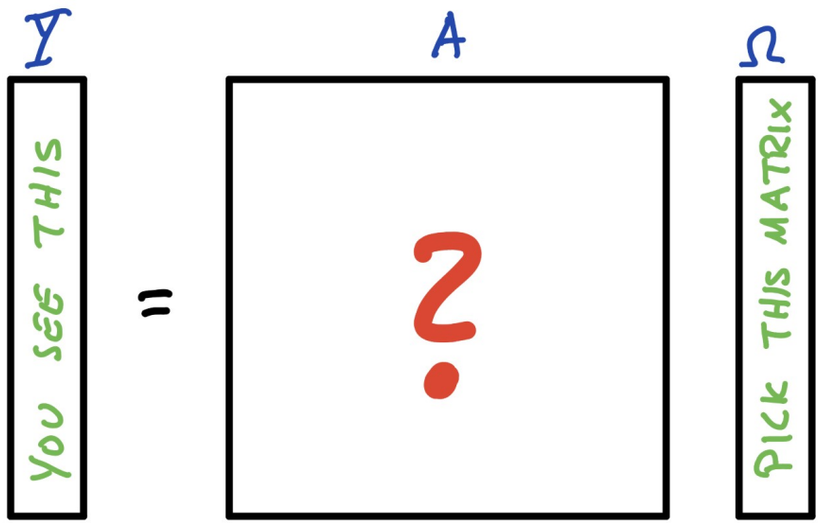
Image credit: Ambikasaran & Darve, arxiv.org #1407.1572

Approximation of rank structured matrices

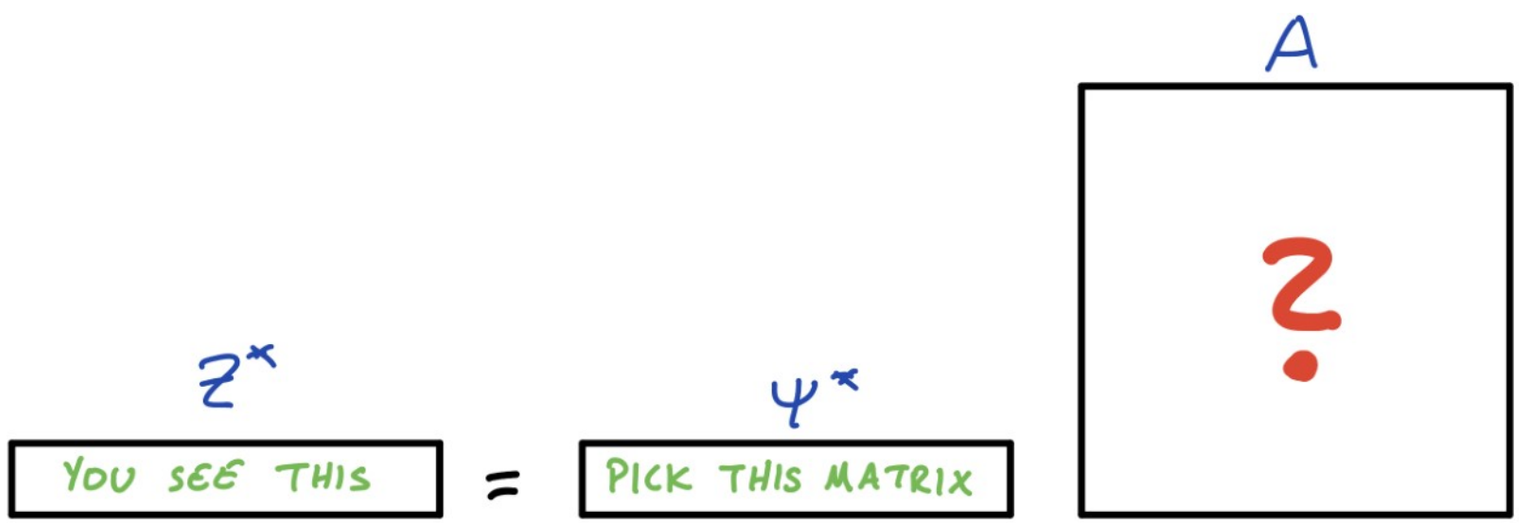
Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ (to be precise, $\mathbf{A}$ is HBS/HSS of rank k). We assume that we can evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$ fast.

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 $\mathbf{A} \in \mathbb{R}^{N \times N}$ (to be precise, $\mathbf{A}$ is HBS/HSS of rank k). 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 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.)

Approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ (to be precise, $\mathbf{A}$ is HBS/HSS of rank k). 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 $\mathbf{Z} = \mathbf{A}^*\mathbf{\Psi}$?

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 lets us overcome key bottlenecks (e.g. LU factorization of a "finite element" matrix is accelerated from $O(N^2)$ to close to linear complexity.)

Approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ (to be precise, $\mathbf{A}$ is HBS/HSS of rank k). 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, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, JCP 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, arxiv arXiv: arXiv:2205.03406, 2022.]*

Approximation of rank structured matrices

Environment: We are given a rank structured matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ (to be precise, $\mathbf{A}$ is HBS/HSS of rank k). 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.

However, a fully black box algorithm with true linear complexity and high practical speed is now available:

- J. Levitt & P.G. Martinsson, arxiv arXiv:2205.02990, 2022.

Key points on randomized singular value decomposition (RSVD):

- High practical speed — interacts with **A** only through matrix-matrix multiplication.
- Highly *communication efficient.*

  Acceleration of classical algorithms such as column pivoted QR.

  Particularly efficient for GPUs, out-of-core computing, distributed memory, etc.
- Reduction in complexity from $O(mnk)$ to $O(mn\log k)$ or even less via *structured random embeddings.*
- Single pass algorithms have been developed for *streaming environments.*

  Not possible with deterministic methods!

More specialized topic — 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`

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