Randomized algorithms for linear algebraic computations

Per-Gunnar Martinsson
Dept. of Mathematics & Oden Institute for Computational Sciences and Engineering
University of Texas at Austin

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

Former and current students and postdocs: Ke Chen, Yijun Dong, Abinand Gopal, Nathan Halko, Nathan Heavner, James Levitt, Sergey Voronin, Heather Wilber, Bowei Wu, Anna Yesypenko.

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

Research support by:

Outline of talk

1. **Randomized low rank approximation**

   “Randomized singular value decomposition” or “RSVD”.

   Techniques based on randomized embeddings.

   Relatively well established material within numerical linear algebra.

2. **Variations of randomized algorithms for low rank approximation**

   Single pass and streaming algorithms.

   Structured random embeddings.

   Matrix approximation via sampling.

3. **Samples of current research directions**

   Least squares problems and linear system solvers.

   Sketching for data compression.

Low rank approximation — problem formulation:

Let **A** be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll n \leq m$.

We seek to compute approximate factors **E** and **F** such that

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

Low rank approximation — problem formulation:

Let **A** be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll n \leq m$.

We seek to compute approximate factors **E** and **F** such that

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

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.

Observe that from **E** and **F** you can compute approximate singular vectors, find dominant eigenvectors (when $\mathbf{A} = \mathbf{A}^*$), find spanning rows/columns, etc.

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

Low rank approximation — problem formulation:

Let **A** be a given $m \times n$ matrix, and let k be an integer such that $1 \leq k \ll n \leq m$.

We seek to compute approximate factors **E** and **F** such that

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

Existing methods for this task are well established. Textbook methods include:

1. Compute the full singular value decomposition of **A**, and then truncate:
   — Resulting approximation is in many regards ”optimal” — best possible fit.
   — Expensive! Cost is $O(mn^2)$. Good for small n, or “expensive” data.
2. Krylov methods:
   — Standard technique for large sparse matrices.
   — Interacts with **A** only through its action on vectors.
   — Theoretically optimal in important regards.
3. Execute Gram-Schmidt on the columns of **A** (“column pivoted QR”):
   — Simple and practical for medium size dense matrices.
   — Not entirely optimal, but often good enough.
   — Cost is $O(mnk)$ since you can stop after k steps.

These methods work great! But room for improvement in important environments.

Randomized SVD:

Objective: Given an $m \times n$ matrix $\mathbf{A}$, find an approximate rank-k partial SVD:

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

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal. (We assume $k \ll \min(m, n)$.)

(A) *Randomized sketching:*

A.1 Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`

A.2 Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`

A.3 Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

(B) *Deterministic post-processing:*

B.1 Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`

B.2 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')`

B.3 Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

The objective of Stage A is to compute an ON-basis that approximately spans the column space of $\mathbf{A}$. The matrix $\mathbf{Q}$ holds these basis vectors and $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

Randomized SVD:

Objective: Given an $m \times n$ matrix $\mathbf{A}$, find an approximate rank-k partial SVD:

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

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal. (We assume $k \ll \min(m, n)$.)

(A) *Randomized sketching:*

A.1 Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`

A.2 Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`

A.3 Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

(B) *Deterministic post-processing:*

B.1 Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`

B.2 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')`

B.3 Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

The objective of Stage A is to compute an ON-basis that approximately spans the column space of $\mathbf{A}$. The matrix $\mathbf{Q}$ holds these basis vectors and $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

Stage B is exact: $\|\mathbf{A} - \mathbf{Q}\underbrace{\mathbf{Q}^*\mathbf{A}}_{=\mathbf{B}}\| = \|\mathbf{A} - \mathbf{Q}\underbrace{\mathbf{B}}_{=\hat{\mathbf{U}}\mathbf{D}\mathbf{V}^*}\| = \|\mathbf{A} - \underbrace{\mathbf{Q}\hat{\mathbf{U}}}_{=\mathbf{U}}\mathbf{D}\mathbf{V}^*\| = \|\mathbf{A} - \mathbf{UDV}^*\|$.

Randomized SVD:

Objective: Given an $m \times n$ matrix $\mathbf{A}$, find an approximate rank-k partial SVD:

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

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal. (We assume $k \ll \min(m, n)$.)

(A) *Randomized sketching:*

A.1 Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`

A.2 Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`

A.3 Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

(B) *Deterministic post-processing:*

B.1 Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`

B.2 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')`

B.3 Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

We claim that the columns of $\mathbf{Y}$ form a good approximate basis for $\text{ran}(\mathbf{A})$.

Observe that $\text{ran}(\mathbf{Y}) \subseteq \text{ran}(\mathbf{A})$ automatically.

Loss of accuracy can happen if $\text{ran}(\mathbf{Y})$ does not capture important directions.

To avoid this, we draw p extra samples, for, say, $p = 5$ or $p = 10$.

Randomized SVD:

Objective: Given an $m \times n$ matrix $\mathbf{A}$, find an approximate rank-k partial SVD:

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

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal. (We assume $k \ll \min(m, n)$.)

(A) *Randomized sketching:*

A.1 Draw an $n \times k$ Gaussian random matrix $\mathbf{\Omega}$. `Omega = randn(n,k)`

A.2 Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A\Omega}$. `Y = A * Omega`

A.3 Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\text{ran}(\mathbf{Q}) = \text{ran}(\mathbf{Y})$. `[Q, ~] = qr(Y)`

(B) *Deterministic post-processing:*

B.1 Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`

B.2 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')`

B.3 Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Important: You only need to ensure that you do not undersample.

Over-sampling is unproblematic, since excess data gets "filtered out" in Stage B.

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

- It is simple to adapt the scheme to the situation where the *tolerance is given,* and the rank has to be determined adaptively.

- Observe how simple the interaction with $\mathbf{A}$ is.

  Two matrix-matrix products only.

  This often leads to very high practical execution speed on modern hardware.

- Accuracy of the basic scheme is good when the singular values decay reasonably fast. When they do not, the scheme can be combined with Krylov-type ideas:

  Taking one or two steps of subspace iteration vastly improves the accuracy.

  For instance, use the sampling matrix $\mathbf{Y} = \mathbf{AA}^*\mathbf{A\Omega}$ instead of $\mathbf{Y} = \mathbf{A\Omega}$.

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

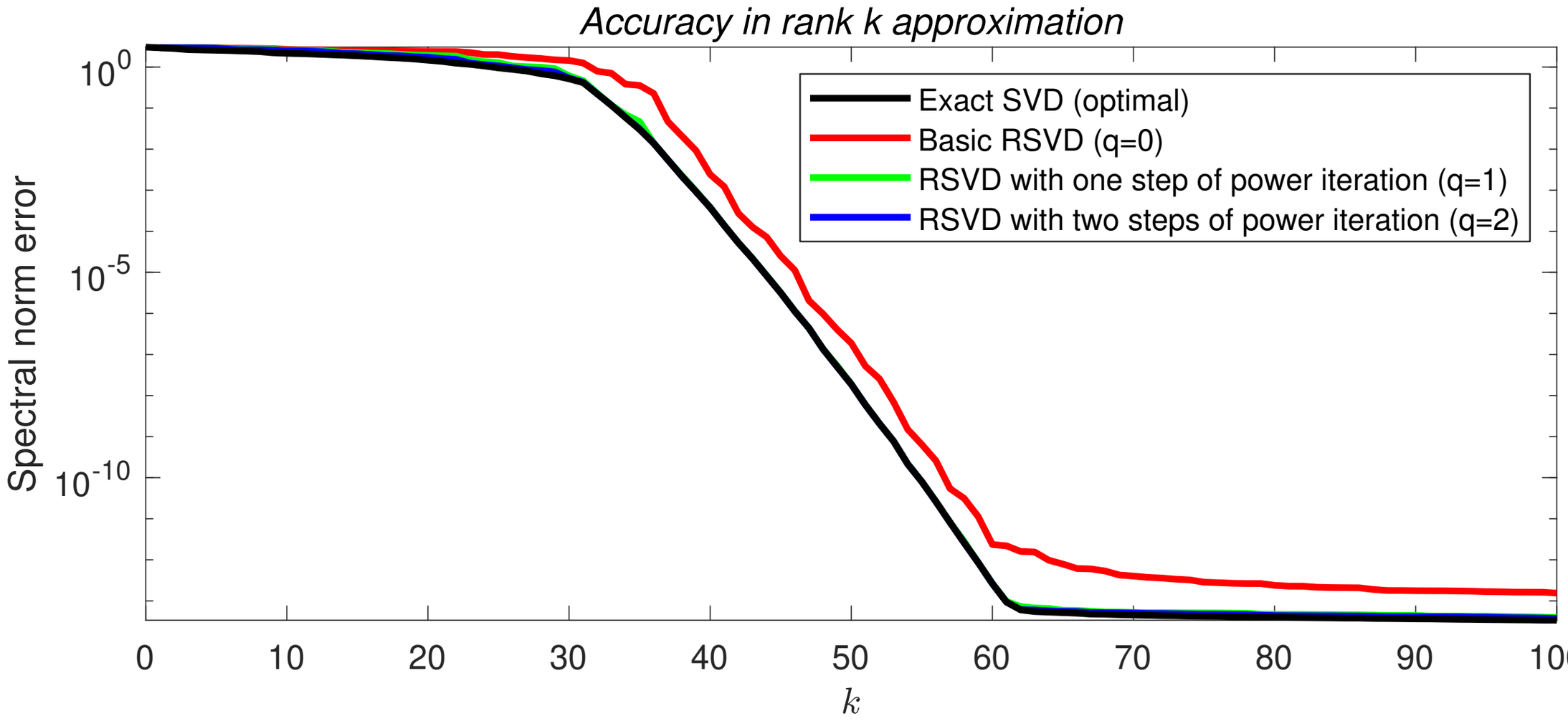
Let us next investigate the accuracy of the method.

To illustrate the errors, we set $p = 0$ (no over-sampling), and then define

$$e_k = \|\mathbf{A} - \mathbf{UDV}^*\| = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|.$$

Eckart-Young theorem: $e_k \geq \sigma_{k+1}$, where σ_j is the j'th singular value of $\mathbf{A}$.

Randomized SVD:



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

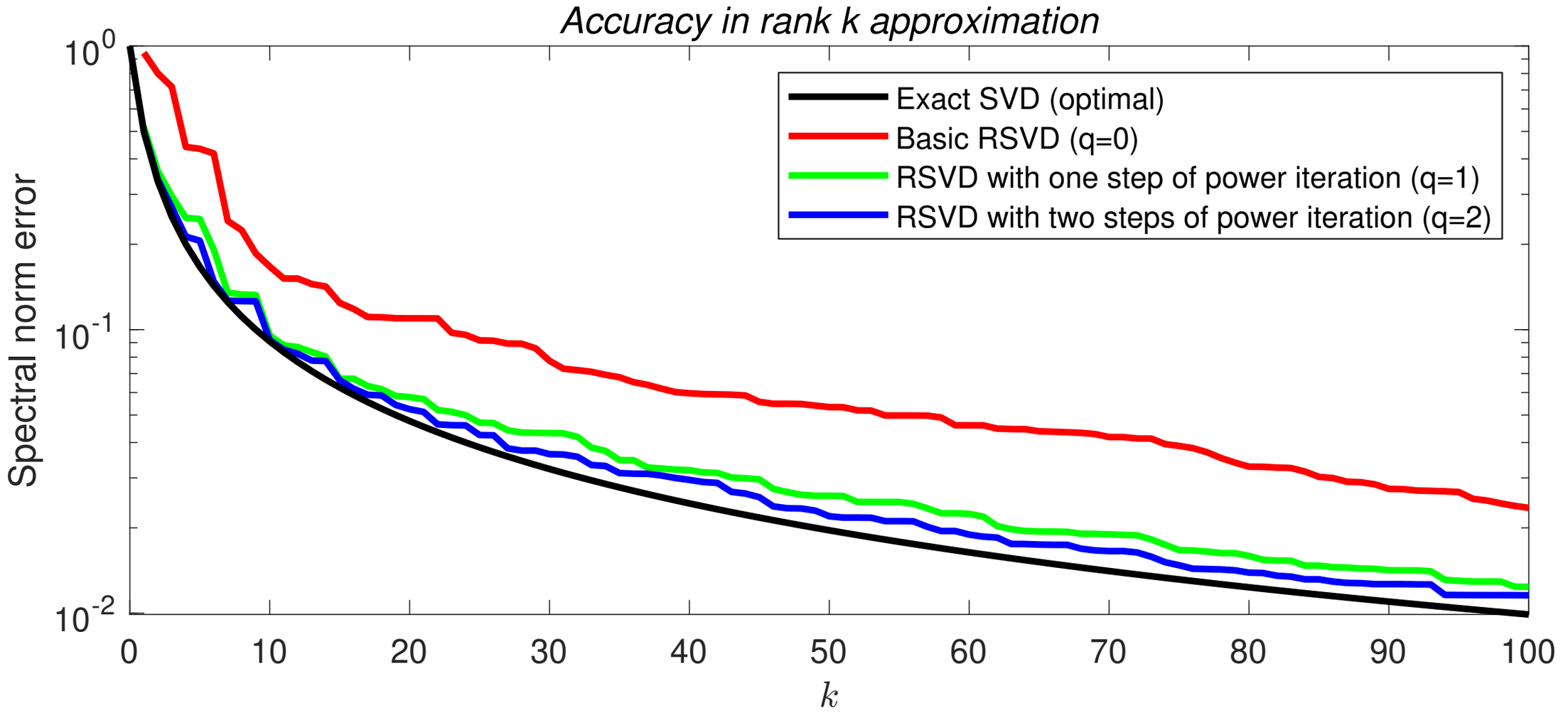
$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

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

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

and where $\mathbf{\Omega}$ is a Gaussian random matrix. (For clarity, no oversampling is done.) The matrix $\mathbf{A}$ is an approximation to a scattering operator for a Helmholtz problem.

Randomized SVD:



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

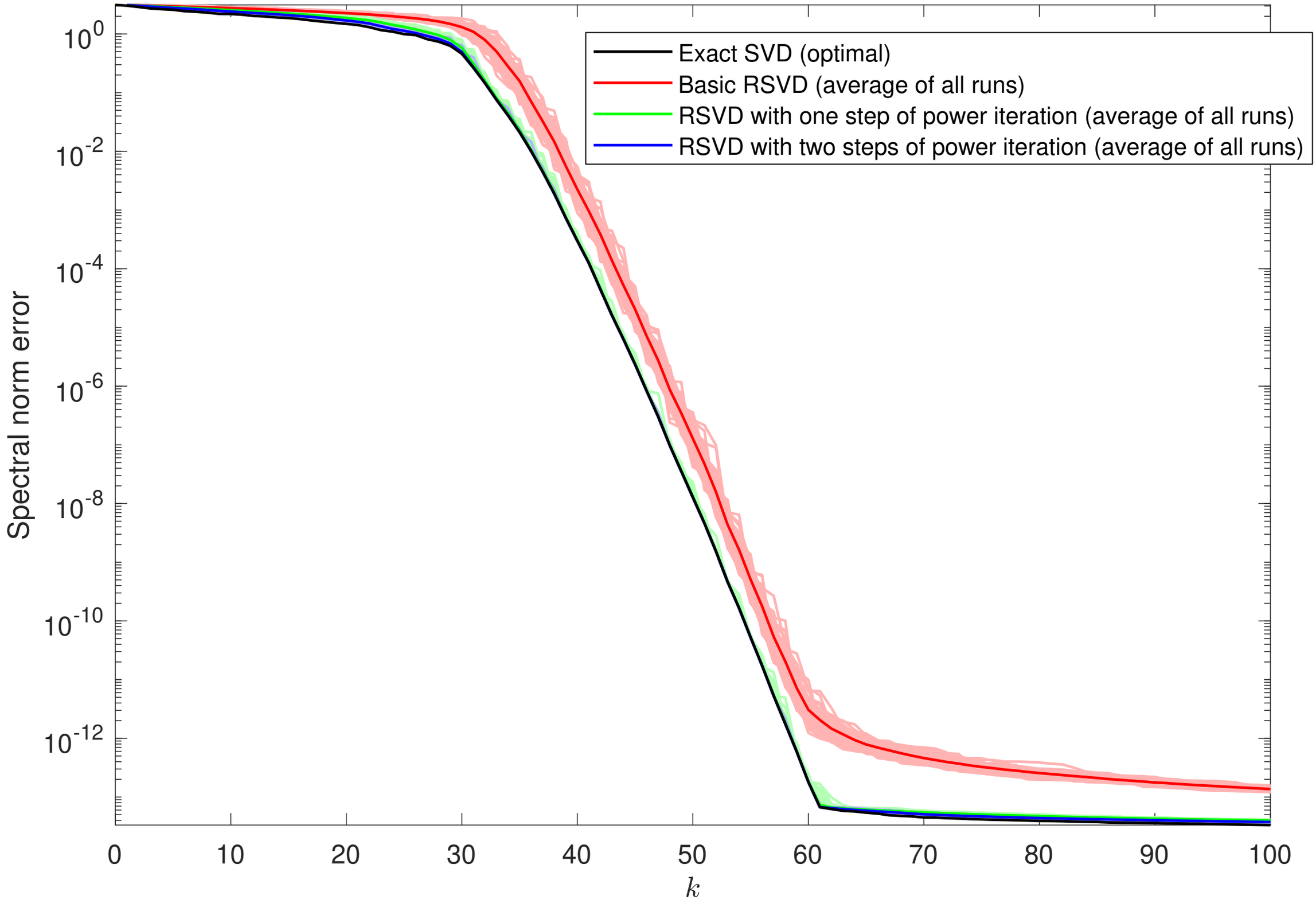
$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

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

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

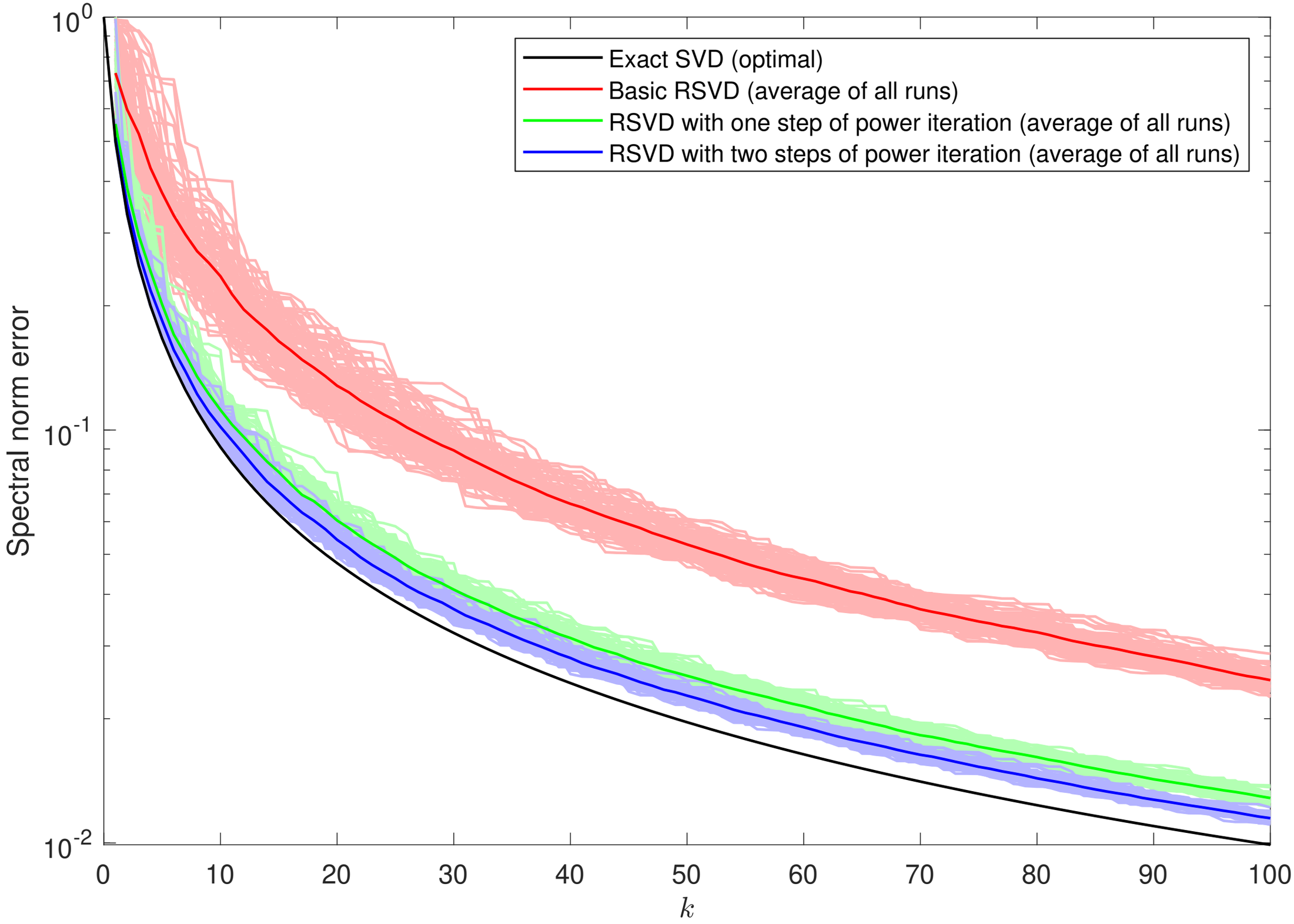
and where $\mathbf{\Omega}$ is a Gaussian random matrix. (For clarity, no oversampling is done.) The matrix $\mathbf{A}$ now has singular values that decay slowly.

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



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

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



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

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

Since the error in the RSVD is a random variable (it depends on the draw of $\mathbf{\Omega}$), any theoretical analysis needs to describe the *probability distribution* of the error.

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

Since the error in the RSVD is a random variable (it depends on the draw of $\boldsymbol{\Omega}$), any theoretical analysis needs to describe the *probability distribution* of the error.

For instance, we can bound the expectation of the error:

Theorem: 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 $\boldsymbol{\Omega}$ be a Gaussian random matrix of size $n \times (k+p)$ and set $\mathbf{Q} = \texttt{orth}(\mathbf{A}\boldsymbol{\Omega})$. Then the average error satisfies

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

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

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

Since the error in the RSVD is a random variable (it depends on the draw of $\boldsymbol{\Omega}$), any theoretical analysis needs to describe the *probability distribution* of the error.

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!)

References (very incomplete!!):

- Martinsson, Rokhlin, Tygert, *Yale-CS-1361*, 2006.
- Halko, Martinsson, Tropp, *SIREV*, 2011. Survey, focus on RSVD.
- 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. Survey. Broader perspective.
- Nakatsukasa, *arXiv:2009.11392*, 2020. Generalized Nyström method.

Outline of talk

1. **Randomized low rank approximation** *[Done!]*

   "Randomized singular value decomposition" or "RSVD".

   Techniques based on randomized embeddings.

   Relatively well established material within numerical linear algebra.

2. **Variations of randomized algorithms for low rank approximation**

   Single pass and streaming algorithms.

   Structured random embeddings.

   Matrix approximation via sampling.

3. **Samples of current research directions**

   Least squares problems and linear system solvers.

   Sketching for data compression.

Randomized SVD: A reformulation

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

Randomized SVD: A reformulation

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

Algorithm: (Merely a restatement of what has already been described.)

Fix an over-sampling parameter p, say $p = 5$.

Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times (k+p)}$.

Form the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.

Orthonormalize the columns of $\mathbf{Y}$ to form an ON matrix $\mathbf{Q}$.

Then $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

Randomized SVD: A reformulation

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

Algorithm: (Merely a restatement of what has already been described.)

Fix an over-sampling parameter p, say $p = 5$.

Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times (k+p)}$.

Form the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.

Orthonormalize the columns of $\mathbf{Y}$ to form an ON matrix $\mathbf{Q}$.

Then $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

We seek to eliminate the intermediate matrices $\mathbf{Y}$ and $\mathbf{Q}$ from the description.

We will use the notion of a *Moore-Penrose pseudoinverse*. Brief recap:

Let $\mathbf{X}$ be an $m \times n$ matrix of rank k, with singular value decomposition

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

Then the *Moore-Penrose pseudoinverse* is

$$\begin{array}{ccccc} \mathbf{X}^\dagger & = & \mathbf{V} & \mathbf{D}^{-1} & \mathbf{U}^*. \\ n \times m & & n \times k & k \times k & k \times m \end{array}$$

The matrix $\mathbf{X}^\dagger$ is a generalization of an inverse for an invertible matrix.

The property we need is: $\mathbf{XX}^\dagger$ is the orthogonal projection onto col($\mathbf{X}$).

Randomized SVD: A reformulation

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

Algorithm: (Merely a restatement of what has already been described.)

Fix an over-sampling parameter p, say $p = 5$.

Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times (k+p)}$.

Form the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.

Orthonormalize the columns of $\mathbf{Y}$ to form an ON matrix $\mathbf{Q}$.

Then $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Using the notion of a *Moore-Penrose pseudoinverse*, we can eliminate the intermediate matrices $\mathbf{Y}$ and $\mathbf{Q}$ from the description:

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

Key claim: The columns of $\mathbf{A\Omega}$ approximately span the column space of $\mathbf{A}$.

Randomized SVD: A reformulation

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

Algorithm: (Merely a restatement of what has already been described.)

Fix an over-sampling parameter p, say $p = 5$.

Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times (k+p)}$.

Form the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.

Orthonormalize the columns of $\mathbf{Y}$ to form an ON matrix $\mathbf{Q}$.

Then $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

Using the notion of a *Moore-Penrose pseudoinverse*, we can eliminate the intermediate matrices $\mathbf{Y}$ and $\mathbf{Q}$ from the description:

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A} = \mathbf{YY}^\dagger\mathbf{A} = (\mathbf{A\Omega})\,(\mathbf{A\Omega})^\dagger\mathbf{A}.$$

Key claim: The columns of $\mathbf{A\Omega}$ approximately span the column space of $\mathbf{A}$.

Next, let us also approximate the row space: Draw a $(k+p) \times m$ Gaussian matrix $\mathbf{\Psi}$, then the rows of $\mathbf{\Psi A}$ approximately span the row space of $\mathbf{A}$. Then:

$$\mathbf{A} \approx (\mathbf{A\Omega})\,(\mathbf{A\Omega})^\dagger\mathbf{A}(\mathbf{\Psi A})^\dagger\,(\mathbf{\Psi A}).$$

Randomized SVD: A reformulation

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

Algorithm: (Merely a restatement of what has already been described.)

Fix an over-sampling parameter p, say $p = 5$.

Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{m \times (k+p)}$.

Form the sample matrix $\mathbf{Y} = \mathbf{A\Omega}$.

Orthonormalize the columns of $\mathbf{Y}$ to form an ON matrix $\mathbf{Q}$.

Then $\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A}$.

Using the notion of a *Moore-Penrose pseudoinverse*, we can eliminate the intermediate matrices $\mathbf{Y}$ and $\mathbf{Q}$ from the description:

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{A} = \mathbf{YY}^\dagger\mathbf{A} = (\mathbf{A\Omega})\,(\mathbf{A\Omega})^\dagger\mathbf{A}.$$

Key claim: The columns of $\mathbf{A\Omega}$ approximately span the column space of $\mathbf{A}$.

Next, let us also approximate the row space: Draw a $(k+p) \times m$ Gaussian matrix $\mathbf{\Psi}$, then the rows of $\mathbf{\Psi A}$ approximately span the row space of $\mathbf{A}$. Then:

$$\mathbf{A} \approx (\mathbf{A\Omega})\,(\mathbf{A\Omega})^\dagger\mathbf{A}(\mathbf{\Psi A})^\dagger\,(\mathbf{\Psi A}).$$

Simplify: $\mathbf{A} \approx (\mathbf{A\Omega})\,(\mathbf{A\Omega})^\dagger\mathbf{A}(\mathbf{\Psi A})^\dagger\,(\mathbf{\Psi A}) = \cdots = (\mathbf{A\Omega})\,\big(\mathbf{\Psi A\Omega}\big)^\dagger\,(\mathbf{\Psi A}).$

Randomized SVD: Double-sided approximation ("generalized Nyström")

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

Randomized SVD: Double-sided approximation ("generalized Nyström")

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

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

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

Observation 1:

The matrix $\mathbf{A}_{\text{approx}}$ 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 for the row and column 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 ("generalized Nyström")

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

Observation 2:

Using Gaussian random matrices, evaluating $\mathbf{A\Omega}$ and $\mathbf{\Psi A}$ 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 ("generalized Nyström")

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

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 ("generalized Nyström")

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

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

Form ap

Observ

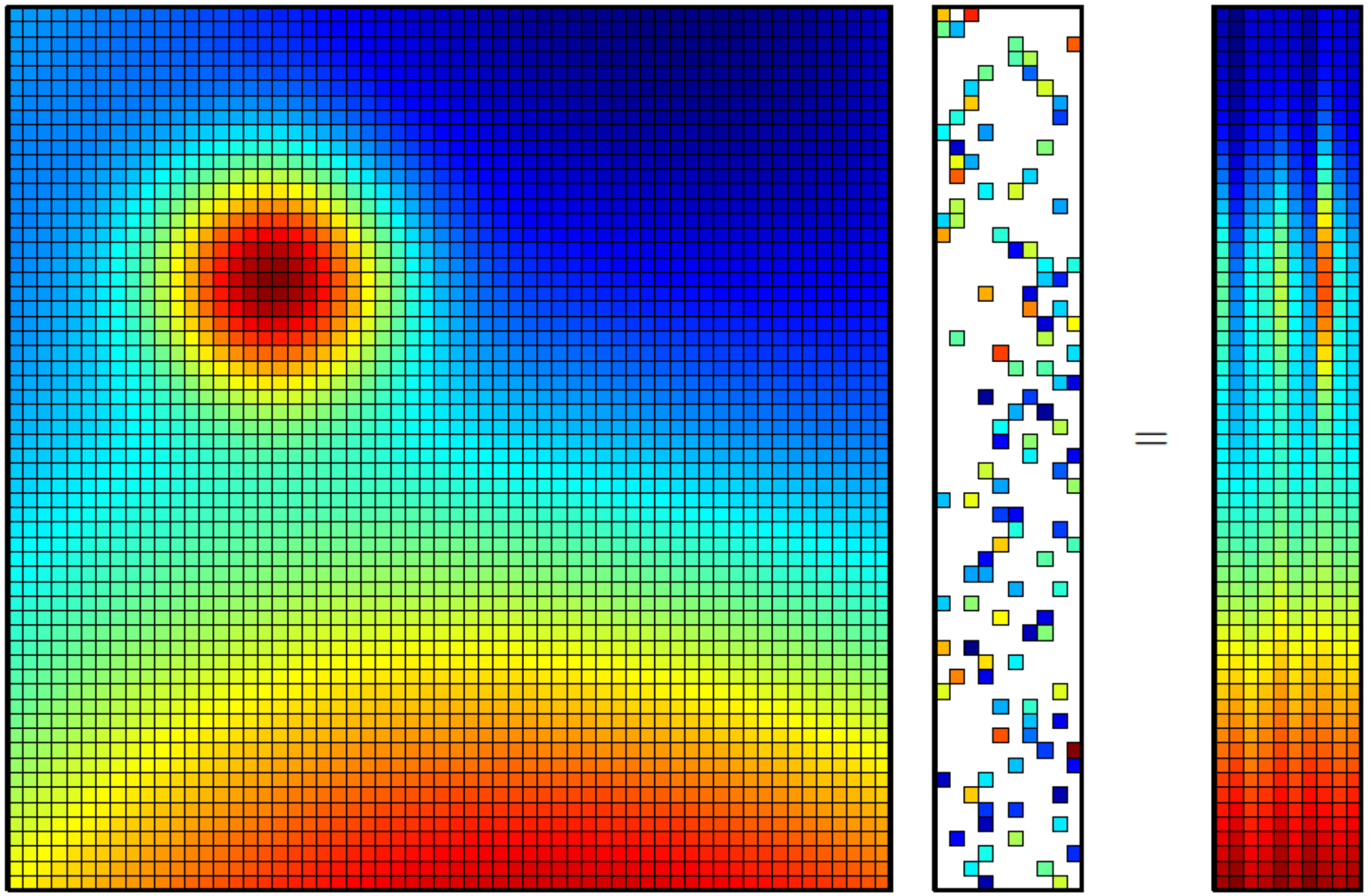
Using G

Instead

with the

- Rar
- Cha
- "Spa

Cos



A

=

Ω

AΩ

n.

ps.

s **Ψ** and **Ω**

flops!

$n \log(k)$).

or $r = 4$.)

The matrix **Ω** *is a sparse random matrix. Two nonzero entries are placed randomly in each row. In consequence, each column of* **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 ("generalized Nyström")

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

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.

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."

Low rank approximation via sampling:

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

Question: Can we be even *more* aggressive? Complexity *less* than $O(mn)$?

Low rank approximation via sampling:

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

Question: Can we be even *more* aggressive? Complexity *less* than $O(mn)$?

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 $\mathbf{A}_{\text{approx}} = \mathbf{CUR}$ where $\mathbf{U}$ is computed from information in $\mathbf{A}(I, J)$. (It should be an approximation to the optimal choice $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$.)

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.

References: Drineas, Kannan & Mahoney (2006); Drineas, Mahoney & Muthukrishnan (2008); Kannan & Vempala (2017); Drineas & Mahoney (2018); Martinsson & Tropp (2020); ...

Low rank approximation via sampling: Connection to structured embeddings

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

Sampling approach (now single sided again): Draw a subset of k columns $\mathbf{C} = \mathbf{A}(:, J)$ where J is drawn at random. Consider the approximant

$$\mathbf{A}_{\text{approx}} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}.$$

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.

Low rank approximation via sampling: Connection to structured embeddings

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

Sampling approach (now single sided again): Draw a subset of k columns $\mathbf{C} = \mathbf{A}(:, J)$ where J is drawn at random. Consider the approximant

$$\mathbf{A}_{\text{approx}} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}.$$

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!*

Low rank approximation via sampling: Connection to structured embeddings

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

Sampling approach (now single sided again): Draw a subset of k columns $\mathbf{C} = \mathbf{A}(:, J)$ where J is drawn at random. Consider the approximant

$$\mathbf{A}_{\text{approx}} = \mathbf{C}\mathbf{C}^{\dagger}\mathbf{A}.$$

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 $\mathbf{U}$ be a matrix drawn from a uniform distribution on the set of $n \times n$ unitary matrices (the "Haar distribution"). Then form

$$\tilde{\mathbf{A}} = \mathbf{A}\mathbf{U}.$$

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 well behaved sample through

$$\mathbf{C} = \tilde{\mathbf{A}}(:, J) = \mathbf{A}\mathbf{U}(:, J).$$

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

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

Observation: A structured random embedding seeks to mimic the action of $\mathbf{U}(:, J)$.

Finding spanning rows and columns — CUR and interpolatory decompositions

In some applications, finding sets of columns/rows that span the column/row space is the primary task. To illustrate, suppose that we are given an $m \times n$ matrix $\mathbf{A}$, a rank $k < \min(m, n)$, and seek to compute a factorization

$$\underset{m \times n}{\mathbf{A}} \quad \approx \quad \underset{m \times k}{\mathbf{C}} \quad \underset{k \times n}{\mathbf{Z}},$$

where $\mathbf{C} = \mathbf{A}(J, :)$ holds a subset of k columns of $\mathbf{A}$, as before.

This problem can be solved via index sampling using leverage scores, as discussed.

Finding spanning rows and columns — CUR and interpolatory decompositions

In some applications, finding sets of columns/rows that span the column/row space is the primary task. To illustrate, suppose that we are given an $m \times n$ matrix $\mathbf{A}$, a rank $k < \min(m, n)$, and seek to compute a factorization

$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{C}} \underset{k \times n}{\mathbf{Z}},$$

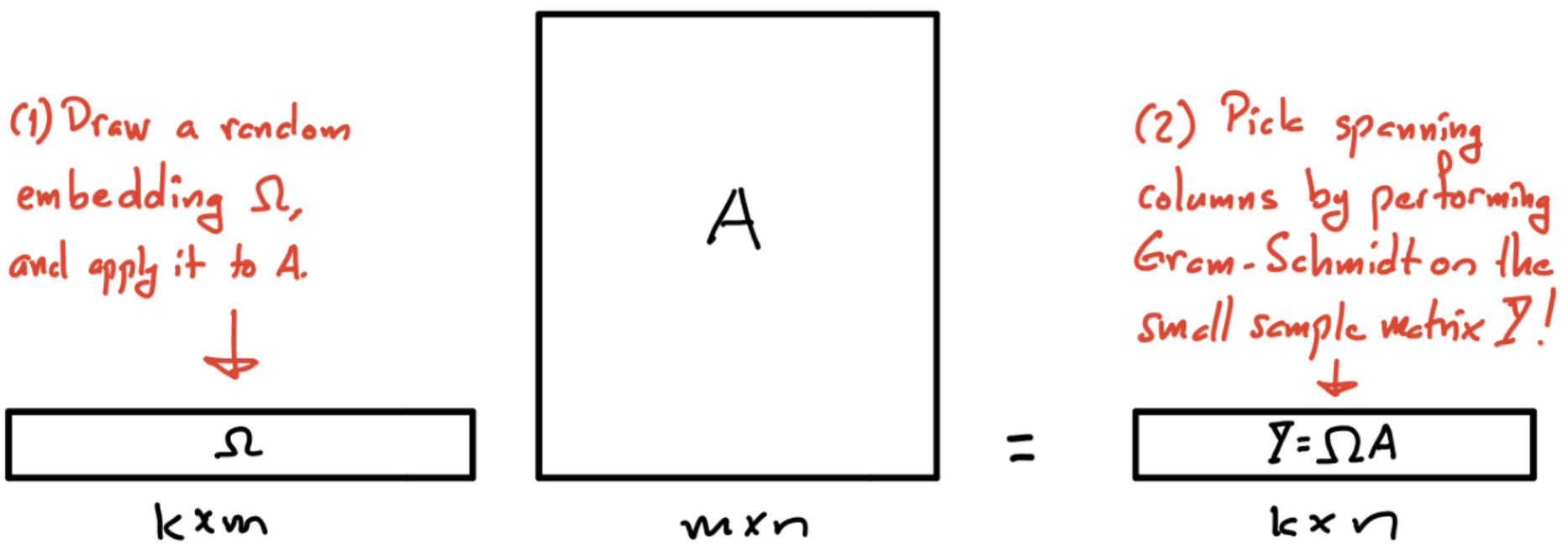
where $\mathbf{C} = \mathbf{A}(J, :)$ holds a subset of k columns of $\mathbf{A}$, as before.

This problem can be solved via index sampling using leverage scores, as discussed.

Alternatively, the following two stage approach works very well:

(1) Apply a random embedding $\mathbf{\Omega}$ to the columns of $\mathbf{A}$ (sparse random is perfect).

(2) Execute a classical pivoting method on $\mathbf{\Omega A}$ (*partially* pivoted LU is ideal).



Finding spanning rows and columns — CUR and interpolatory decompositions

In some applications, finding sets of columns/rows that span the column/row space is the primary task. To illustrate, suppose that we are given an $m \times n$ matrix $\mathbf{A}$, a rank $k < \min(m, n)$, and seek to compute a factorization

$$\underset{m \times n}{\mathbf{A}} \quad \approx \quad \underset{m \times k}{\mathbf{C}} \quad \underset{k \times n}{\mathbf{Z}},$$

where $\mathbf{C} = \mathbf{A}(J, :)$ holds a subset of k columns of $\mathbf{A}$, as before.

This problem can be solved via index sampling using leverage scores, as discussed.

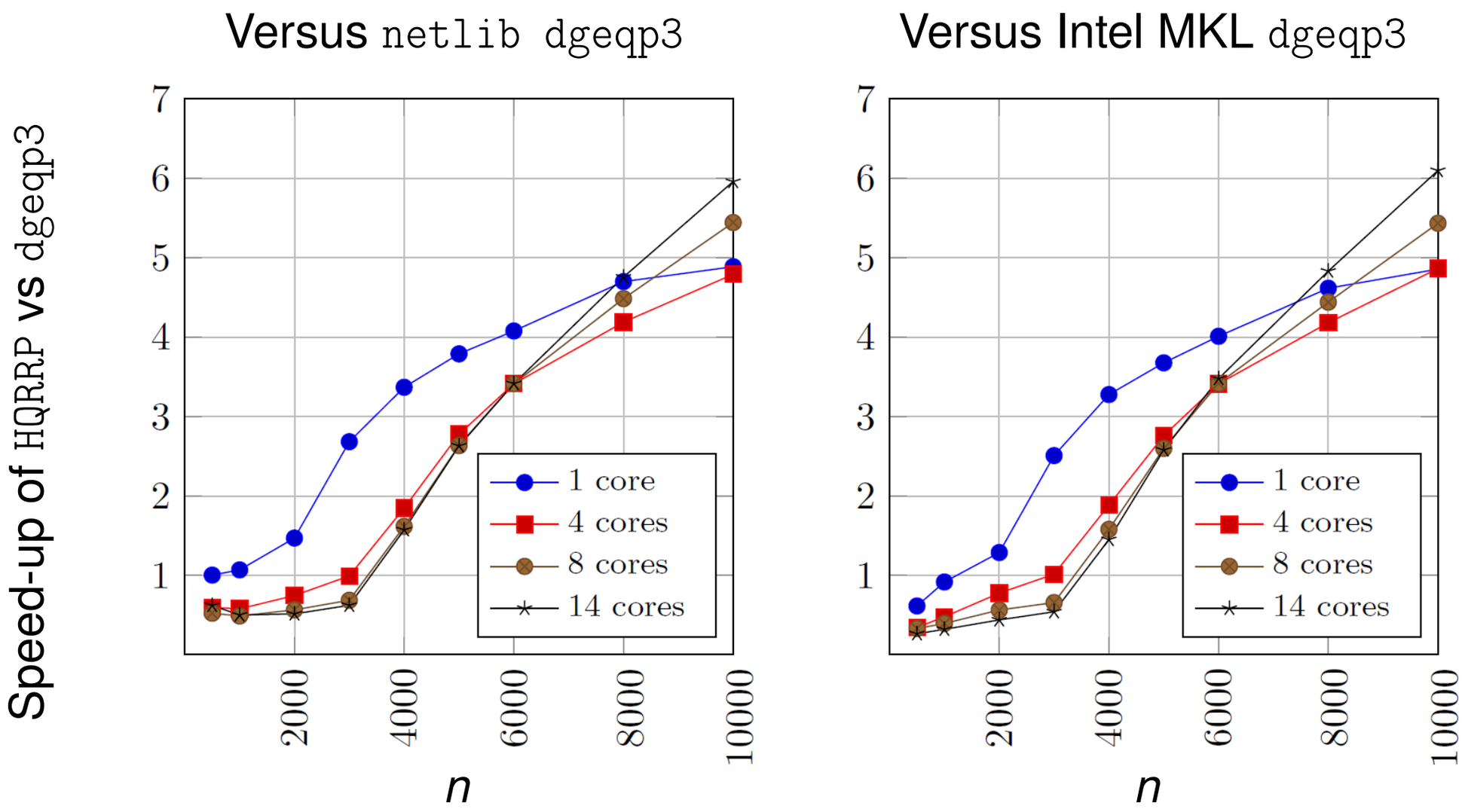
Alternatively, the following two stage approach works very well:

(1) Apply a random embedding $\mathbf{\Omega}$ to the columns of $\mathbf{A}$ (sparse random is perfect).

(2) Execute a classical pivoting method on $\mathbf{\Omega A}$ (*partially* pivoted LU is ideal).

References: Liberty, Woolfe, Martinsson, Rokhlin and Tygert (2007); Sorensen & Embree (2006); Halko, Martinsson, Tropp (2011); Dong & Martinsson, arXiv:2104.05877, 2021; . . .

Computing full factorizations — accelerated column pivoted QR: The column selection strategy on the previous slide has been applied to resolve a classical problem in numerical linear algebra: How do you pick *groups* of pivots when executing column pivoted QR? The purpose is to move flops from BLAS2 to BLAS3 operations.



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.

Outline of talk

1. **Randomized low rank approximation** *[Done!]*

   "Randomized singular value decomposition" or "RSVD".

   Techniques based on randomized embeddings.

   Relatively well established material within numerical linear algebra.

2. **Variations of randomized algorithms for low rank approximation** *[Done!]*

   Single pass and streaming algorithms.

   Structured random embeddings.

   Matrix approximation via sampling.

3. **Samples of current research directions**

   Least squares problems and linear system solvers.

   Sketching for data compression.

Solving least squares problems, linear systems, . . .

- **Overdetermined least squares problems.**

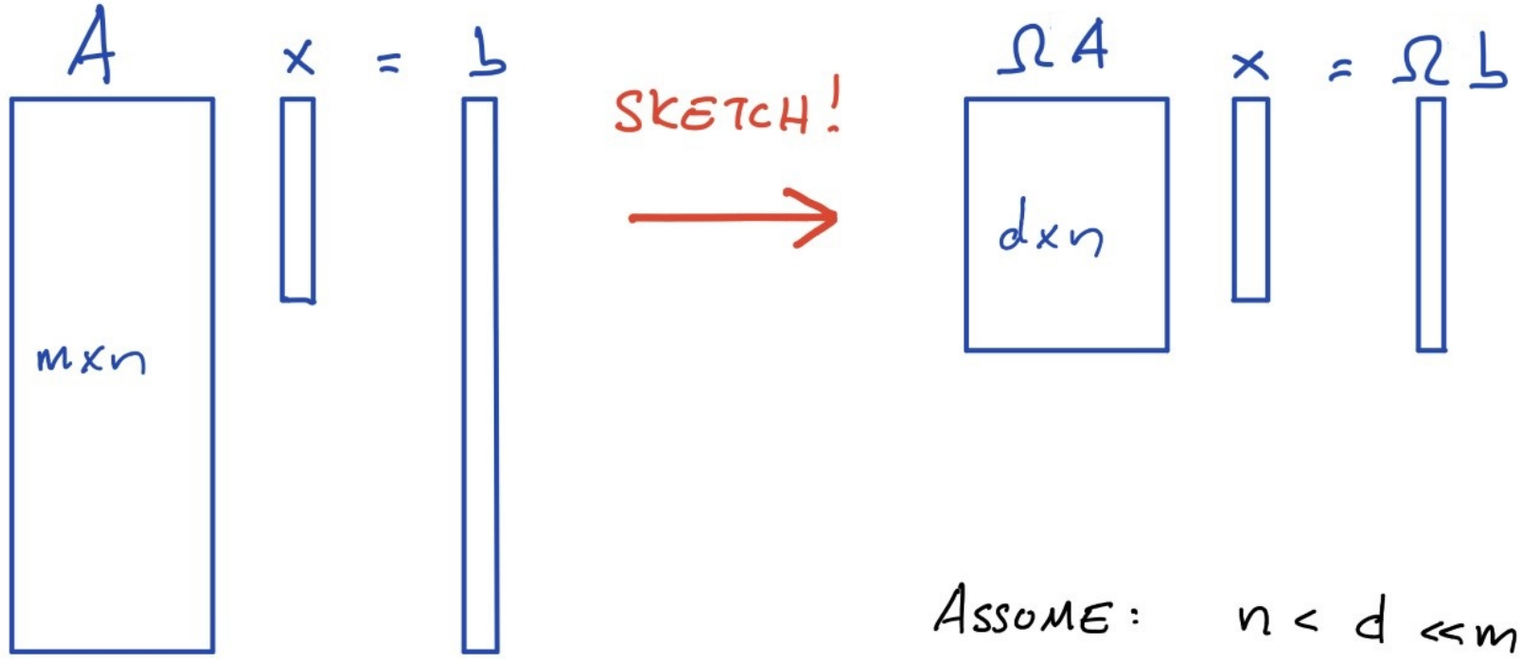
  Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ for $m \gg n$, and that you seek to solve $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$.

Solving least squares problems, linear systems, . . .

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

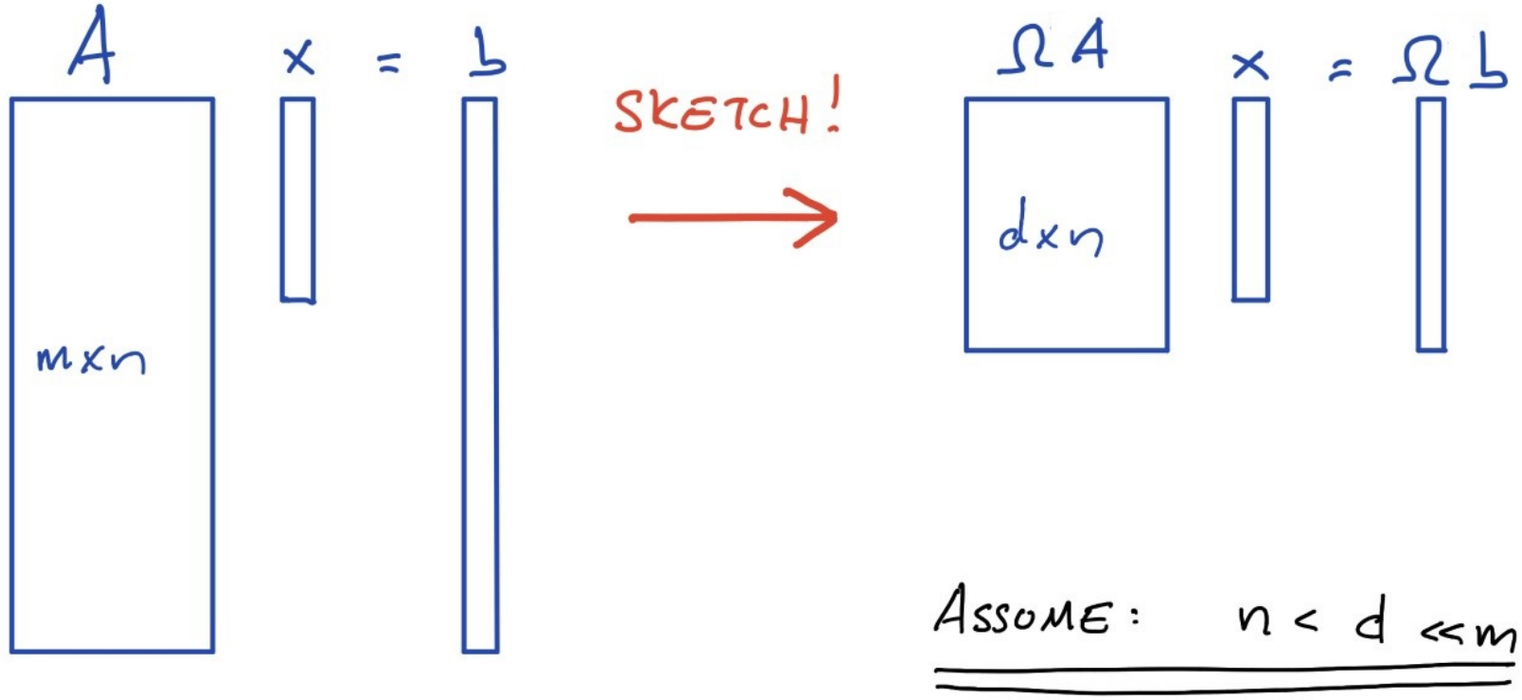


Solving least squares problems, linear systems, . . .

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



A bold approach — "sketch-to-solve":

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

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

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

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

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

Solving least squares problems, linear systems, . . .

- **Overdetermined least squares problems.** *Sketch-to-precondition paradigm.*
- **Randomized Kaczmarz:** Randomization in a classical algorithm. Particularly effective when randomized embeddings are incorporated. *(Strohmer/Vershynnin 2009, Needell/Ward/Srebro 2014, Gower/Richtarik 2015, Liu/Wright 2016, . . .)*
- **Eliminating pivoting in Gaussian elimination:** D. Stott Parker showed in 1995 that you can eschew pivoting in Gaussian elimination if you first "scramble" the coefficient matrix through "pre-conditioning" via random unitary maps. Early example of fast J-L transform! Related work by Demmel/Dumitriu/Holtz (2007).
- **Graph Laplacians:** Linear systems whose coefficient matrix is a graph Laplacian can be solved using randomized methods in close to linear (in the number of edges) complexity. The idea is to compute an approximate Cholesky factorization $\mathbf{A} \approx \mathbf{C}\mathbf{C}^*$, and then use $\mathbf{C}$ as a preconditioner in conjugate gradients. Can be hybridized with ideas from nested dissection to achieve high practical speed for problems in scientific computing. *(Spielman/Teng 2004, Kyng/Sachdeva 2016, Koutis/Miller/Tolliver 2011, Livne/Brandt 2012, Spielman 2020, Chen/Liang/Biros 2020, . . .).*

Solution of huge linear systems via randomized sampling

Randomized sampling has enabled the solution of some systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ that would otherwise be intractable. Think of systems so large that even forming $\mathbf{A}$ is impossible.

In such situations, we must have some à priori information about structure or regularity in $\mathbf{A}$ that can be exploited. A common situation is that $\mathbf{A}(i,j) = k(\mathbf{x}_i, \mathbf{x}_j)$, where k is a given *kernel function*, and $\{\mathbf{x}_i\}_{i=1}^N$ is a given set of points in $\mathbb{R}^d$.

- **Kernel ridge regression:** Need to solve a linear system $\big(\mathbf{A} + \mu\mathbf{I}\big)\mathbf{x} = \mathbf{b}$ where $\mathbf{A}$ is a kernel matrix. Techniques based on Nyström approximation combined with randomized sampling have proven highly effective. *(Alaoui/Mahoney 2015, Avron/Clarkson/Woodruff 2017, Musco/Musco 2017, Rudi/Calandriello/Carratino/Rosasco 2018, . . .)*

- **Rank-structured representations of kernel matrices:** Multifrontal sparse direct solvers; FMMs; boundary integral equation methods; covariance matrices in Gaussian processes; log-determinants; etc. *(Martinsson 2011, March/Xiao/Biros 2015, Ambikasaran/Foreman-Mackey/Greengard/Hogg/O'Neil 2015, Ghysels/Li/Gorman/Rouet 2016, Yu/Levitt/Reiz/Biros 2017, Rebrova/Chávez/Liu/Ghysels/Li 2018, Geoga/Anitescu/Stein 2019, . . .)*

- **Electronic structure calculations:** Impossible even to store $\mathbf{x}$. *(Lim/Weare 2017)*

Randomized embeddings: "Sketching" for data compression

Storage of scientific data is often a core challenge:

- Massive sensor arrays.
- High resolution and multiband imaging.
- Time stepping in scientific simulations — fluid dynamics, molecular dynamics, etc.

In some environments, well-established techniques based on harmonic analysis (Fourier bases, wavelets, etc) are helpful.

When less is known about the data à priori, computing *randomized sketches* has proven to be valuable. Many different regimes have been pursued:

- Low rank matrices.
- Various tensor decompositions.
- Sparse Fourier transforms.

References: *Woolfe/Liberty/Rokhlin/Tygert (2008), Clarkson/Woodruff (2009), Halko/Martinsson/Tropp (2011), Li/Nguyen/Woodruff (2014), Ghashami/Liberty/Phillips/Woodruff (2016), Kressner/Periša (2017), Tropp/Yurtsever/Udell/Cevher (2017–), . . .*

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!

Current research directions:

- High performance implementations.
- Faster algorithms for computing *full* matrix factorizations.
- Solvers for linear systems and for least squares problems.
- Applications of randomized embeddings outside of linear algebra: Faster nearest neighbor search, faster clustering algorithms, data compression on the fly, etc.

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)