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: Chao Chen, 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.

(3) **Samples of current research directions** (time permitting)

Finding spanning row and columns.

Randomized compression of rank structured matrices.

(Matrix approximation via sampling.)

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

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

Low rank approximation — problem formulation:

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

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

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

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 $\mathbf{E}$ and $\mathbf{F}$ you can compute approximate singular vectors, find dominant eigenvectors (when $\mathbf{A}$ is normal), 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

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

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:

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

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 $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `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 $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

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 $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `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 $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

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

Randomized SVD:

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

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

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 $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$. `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 $\boldsymbol{\Omega}$. `Omega = randn(n,k)`

A.2 Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\boldsymbol{\Omega}$. `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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

(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 $\mathbf{A} \approx \mathbf{UDV}^*$.	
(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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.	
(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 $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

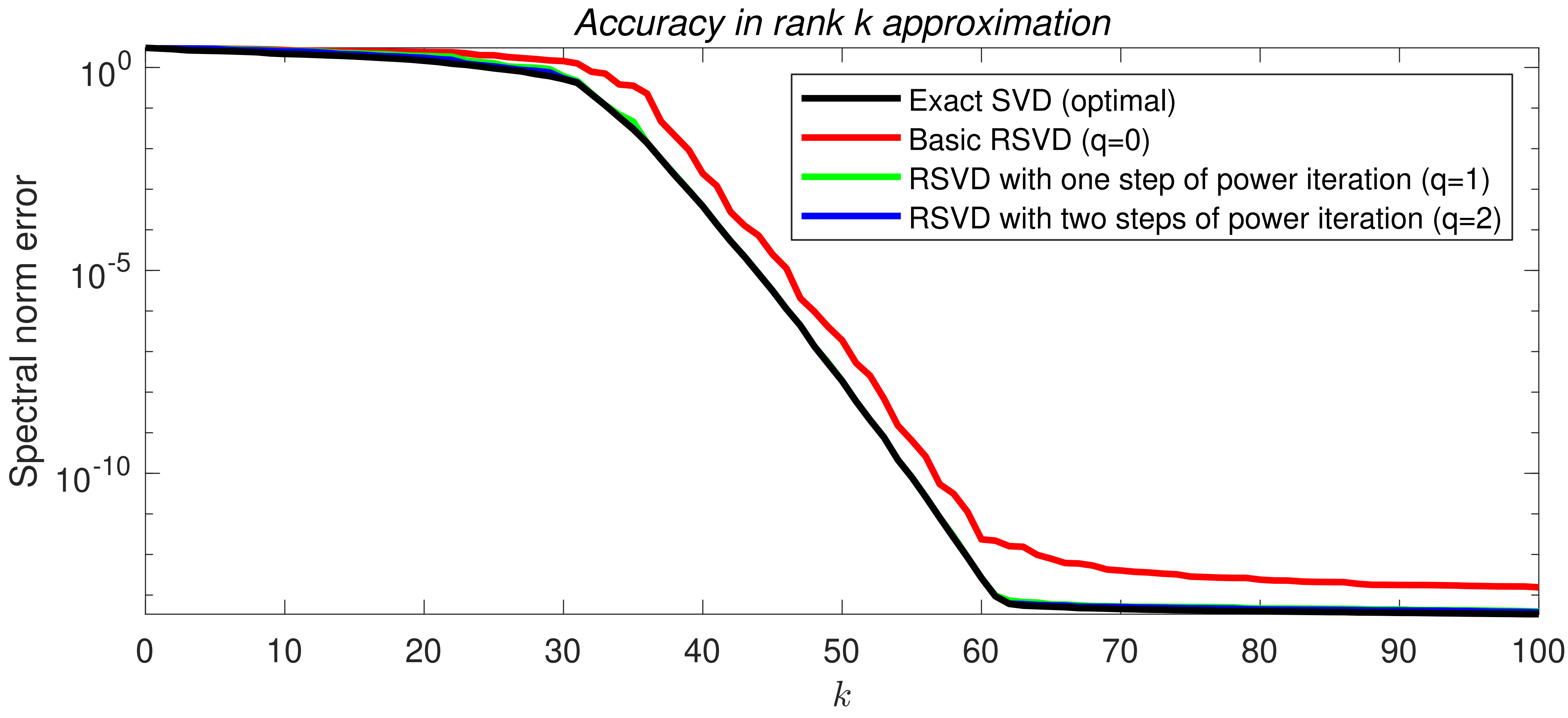
Let us next investigate the accuracy of the method.

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

$$e_k = \|\mathbf{A} - \mathbf{U}\mathbf{D}\mathbf{V}^*\| = \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|.$$

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

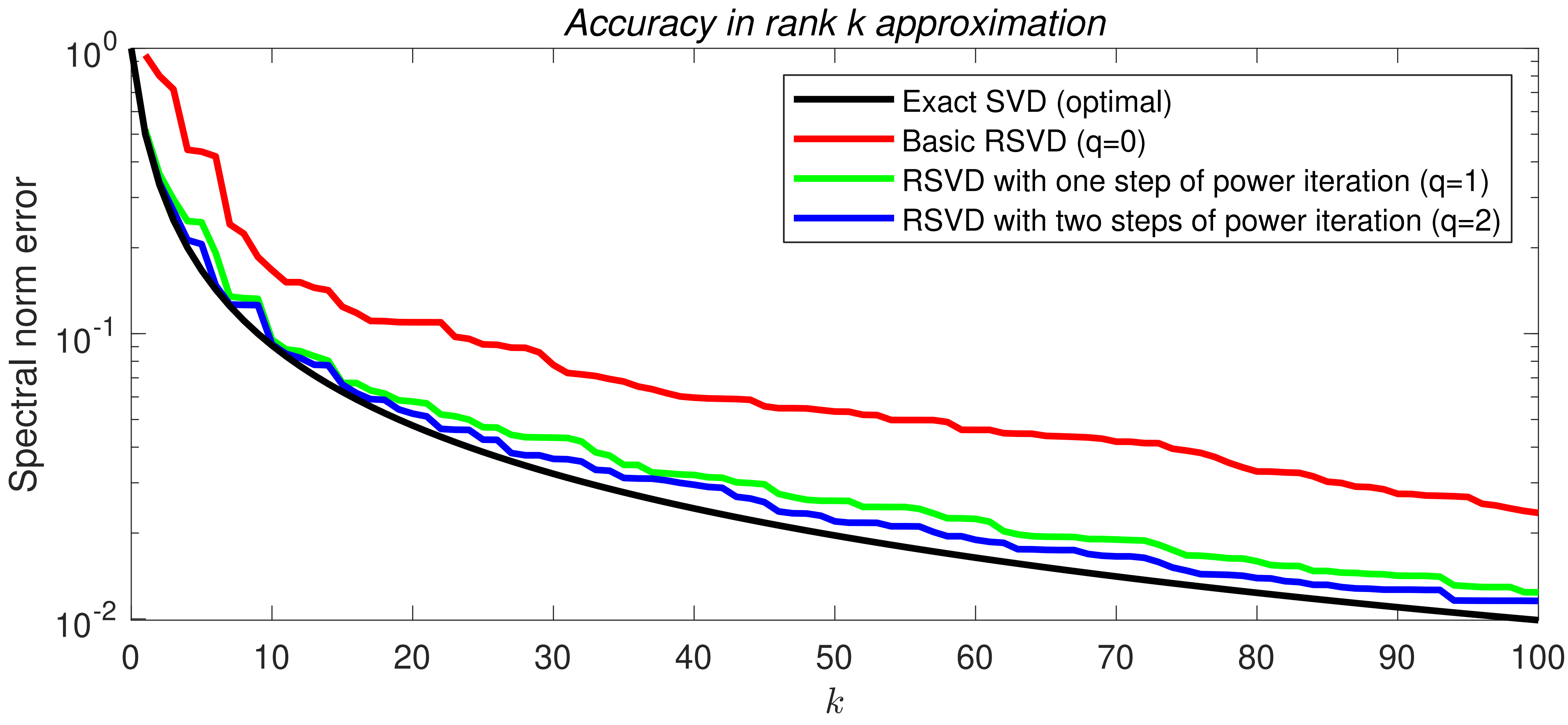
$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

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

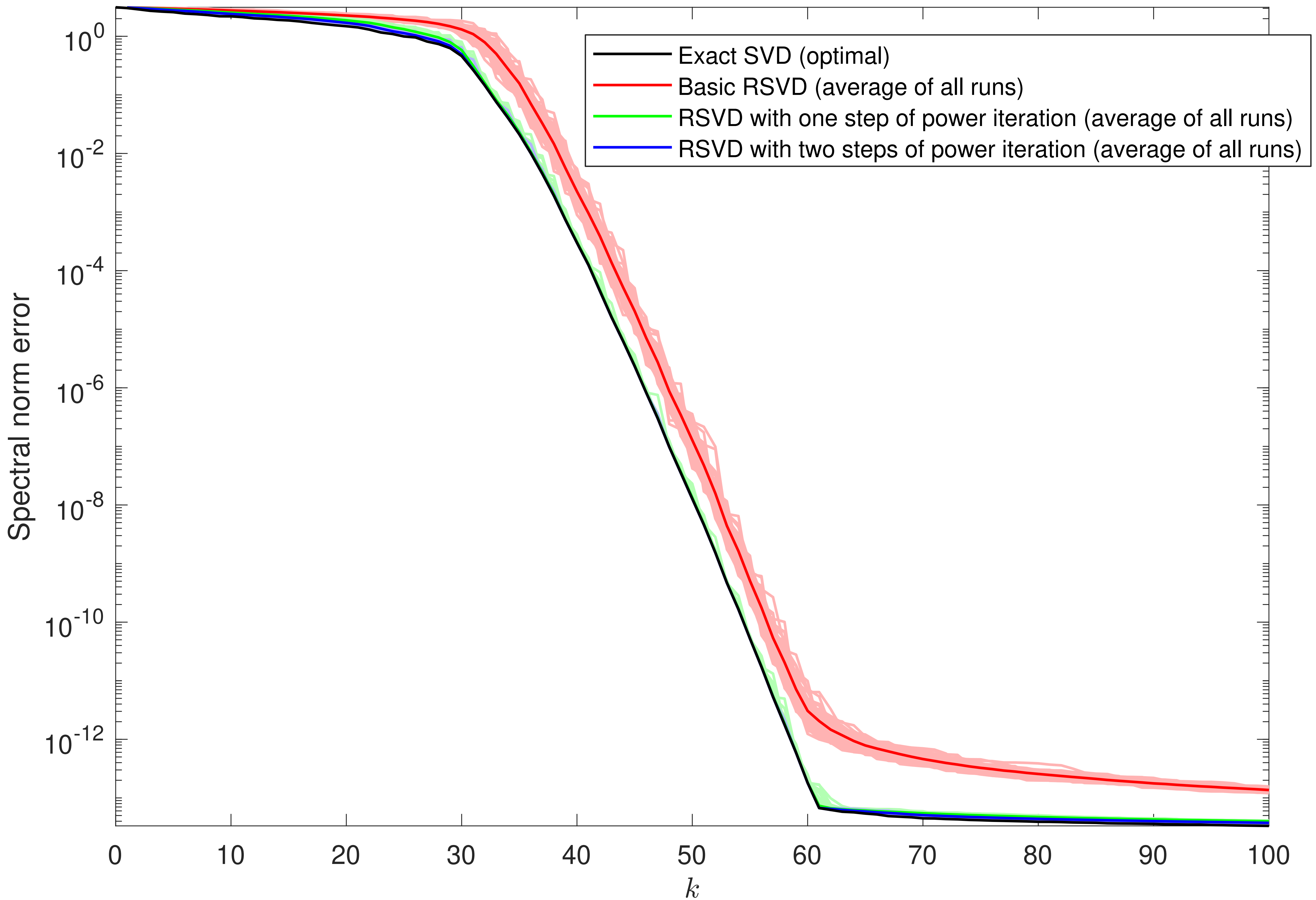
$$e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|,$$

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

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

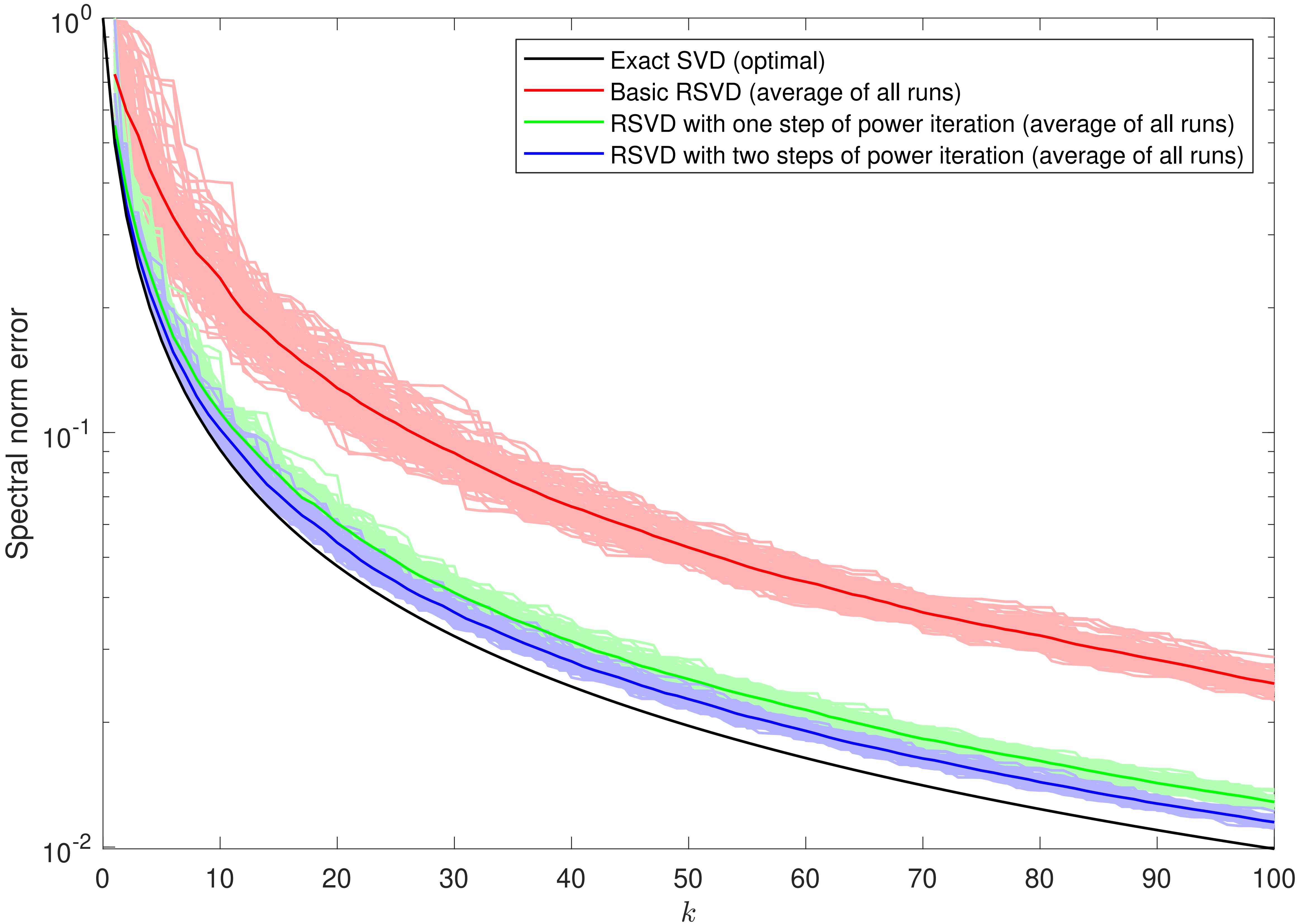
and where $\mathbf{\Omega}$ is a Gaussian random matrix. (For clarity, no oversampling is done.) The matrix $\mathbf{A}$ now has singular values that decay slowly.

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:

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

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:

<table>
<tr><td colspan="2">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 $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.</td></tr>
<tr><td>(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. $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^*\mathbf{Y}$.</td><td>(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 $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.</td></tr>
</table>

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.

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

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

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

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

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.

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.

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.

(3) **Samples of current research directions** (time permitting)

Finding spanning row and columns.

Randomized compression of rank structured matrices.

(Matrix approximation via sampling.)

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

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

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

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

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

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

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: $\boxed{\mathbf{X}\mathbf{X}^\dagger \text{ is the orthogonal projection onto } \mathrm{col}(\mathbf{X}).}$

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

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 $\mathbf{A}$.

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

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:

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

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

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

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:

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

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

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 $\mathbf{\Omega} \in \mathbb{R}^{m\times(k+p)}$ 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 1:

The matrix $\mathbf{A}_{\rm approx}$ 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 1: *Using different over-sampling parameters for the row and column spaces is often better.*

Note 2: *More sophisticated methods exist that draw additional sketches.*

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

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

Observation 2:

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

Instead of Gaussian random matrices, we can use *structured random matrices* $\mathbf{\Psi}$ and $\mathbf{\Omega}$ with the property that $\mathbf{A}\mathbf{\Omega}$ *and* $\mathbf{\Psi}\mathbf{A}$ *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 $\mathbf{A} \approx (\mathbf{A\Omega})\left(\mathbf{\Psi A \Omega}\right)^{\dagger}(\mathbf{\Psi A}) =: \mathbf{A}_{\rm approx}$.

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 $O(mn)$!

Randor

Task: F

Algoritl ... n.

Form ap

Observ

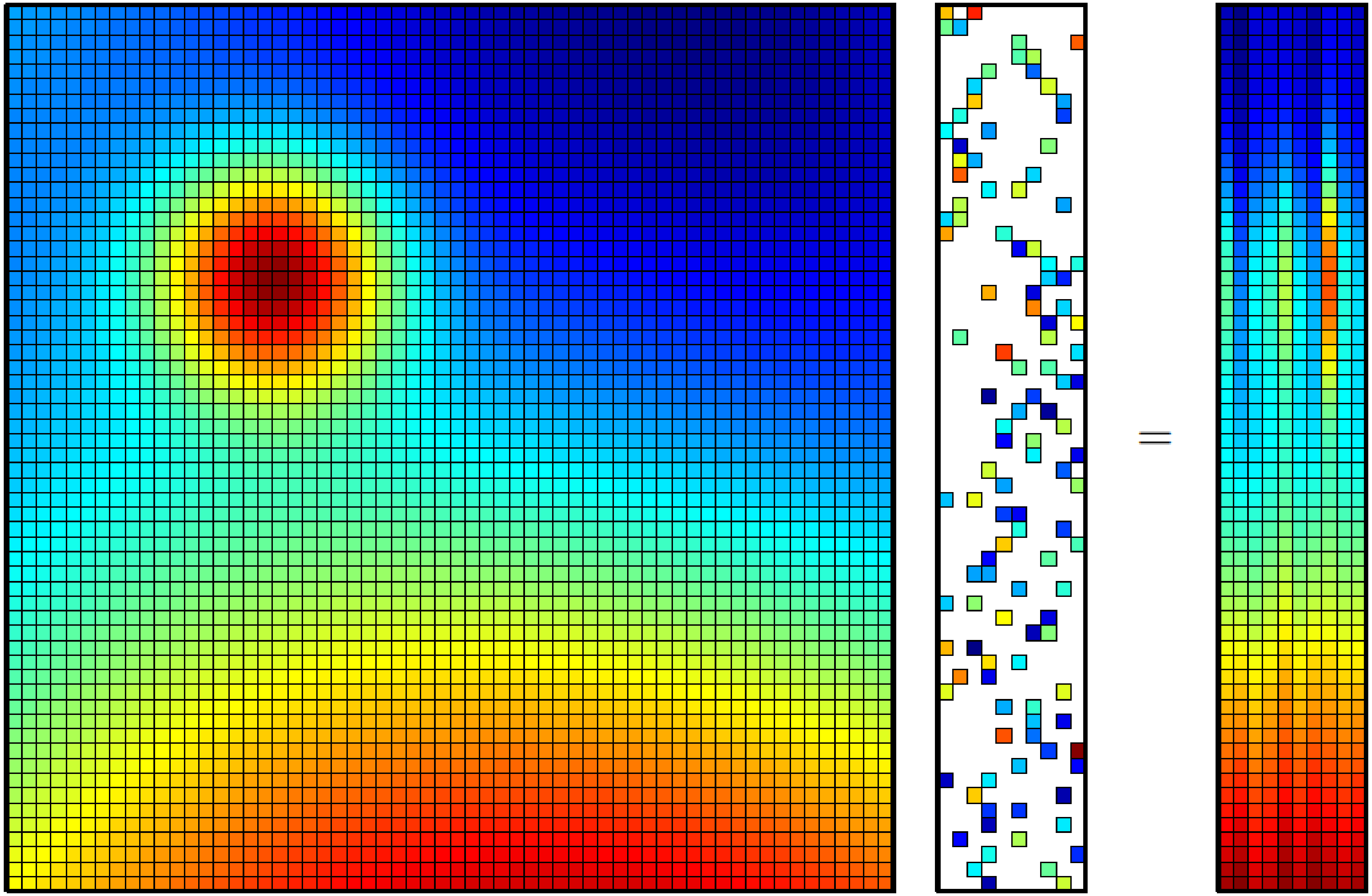
Using G ... ps.

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

with the ... r *flops!*

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

  Cos



A **Ω** **AΩ**

The matrix $\mathbf{\Omega}$ *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 $\mathbf{A} \approx (\mathbf{A\Omega})\,(\mathbf{\Psi A \Omega})^{\dagger}\,(\mathbf{\Psi A}) =: \mathbf{A}_{\mathrm{approx}}$.

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."

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.

(3) **Samples of current research directions** (time permitting)

Finding spanning row and columns.

Randomized compression of rank structured matrices.

(Matrix approximation via sampling.)

Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

A common task in linear algebra is to find sets of columns/rows that span the column/row space of a matrix. 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

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{C} & \mathbf{Z}, \\ m \times n & & m \times k & k \times n \end{array}$$

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

Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

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

$$\begin{array}{cccc} \mathbf{A} & \approx & \mathbf{C} & \mathbf{Z}, \\ m \times n & & m \times k & k \times n \end{array}$$

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

Applications:

- *Data interpretation:* The columns you pick sometimes correspond to specific variables that explain some data set — specific genes, specific stocks, etc.
- *Preserving structure:* If $\mathbf{A}$ is sparse/non-negative, then so is $\mathbf{C}$. Particularly powerful in a "CUR" decomposition $\mathbf{A} \approx \mathbf{CUR}$ where $\mathbf{R}$ holds a subset of rows.
- *Numerical stability:* For matrices that are rank deficient, or nearly rank deficient, factorizations such as column pivoted QR, and fully pivoted LU factorizations are very useful. Finding pivoting strategies is closely related to the problem of finding close to optimal sets of spanning rows/columns.

Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

A common task in linear algebra is to find sets of columns/rows that span the column/row space of a matrix. 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

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{C} & \mathbf{Z}, \\ m \times n & & m \times k & k \times n \end{array}$$

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

Standard techniques:

- Golub-Businger pivoting strategy: The standard tool. Simple, attractive flop count, generally works well, but can be substantially suboptimal. Challenging to implement efficiently on modern hardware.
- Specialized pivoting strategies such as Gu-Eisenstat: Guaranteed to select columns that are close to optimal. Rarely implemented (too complicated).
- Randomized sampling strategies: Very popular subject in theoretical CS. Powerful asymptotic theory. In practice, competitive only for huge matrices, or matrices where entry evaluation is expensive.

Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

A common task in linear algebra is to find sets of columns/rows that span the column/row space of a matrix. 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

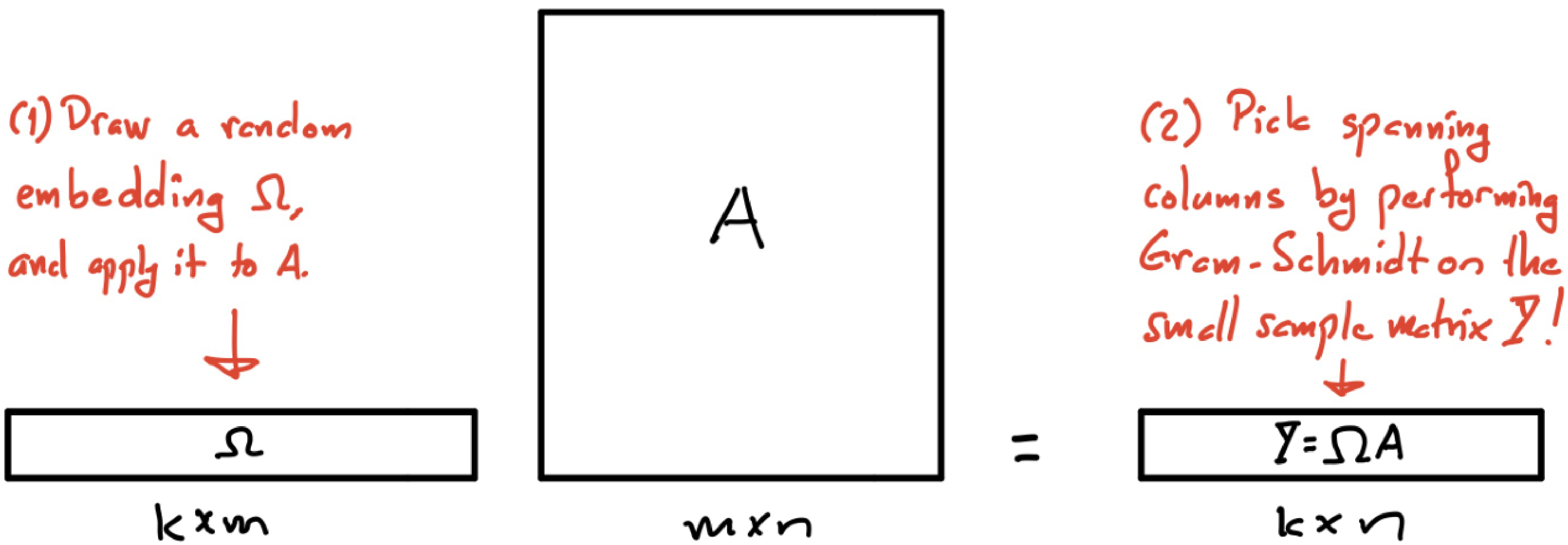
$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{C}} \; \underset{k \times n}{\mathbf{Z}},$$

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

A hybrid randomized / classical approach:

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

(2) Execute a classical pivoting method on $\mathbf{\Omega A}$



Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

A common task in linear algebra is to find sets of columns/rows that span the column/row space of a matrix. 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

$$\begin{array}{cccc} \mathbf{A} & \approx & \mathbf{C} & \mathbf{Z}, \\ m \times n & & m \times k & k \times n \end{array}$$

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

A hybrid randomized / classical approach:

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

(2) Execute a classical pivoting method on $\mathbf{\Omega A}$

- Sparse random embeddings work well for step (1).
- *Partially* pivoted LU can be used for step (2). Fast!! (Very surprising to me!)
- Simple to implement.
- Overall complexity as low as $O(mn + k^2 n)$. (Versus the classical $O(mnk)$.)

Finding spanning rows and columns — CUR/interpolatory/CPQR decompositions

A common task in linear algebra is to find sets of columns/rows that span the column/row space of a matrix. 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

$$\begin{array}{cccc} \mathbf{A} & \approx & \mathbf{C} & \mathbf{Z}, \\ m \times n & & m \times k & k \times n \end{array}$$

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

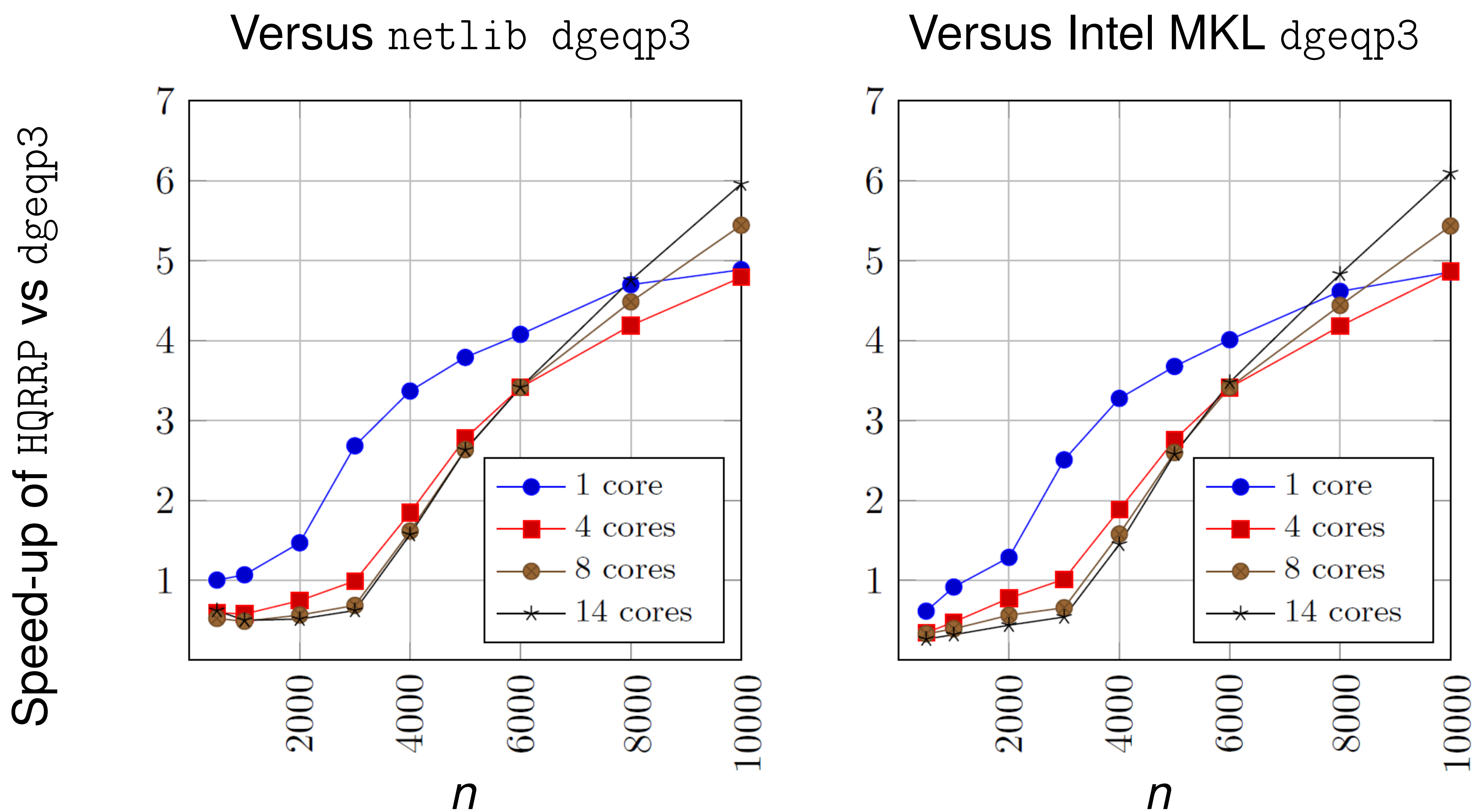
A hybrid randomized / classical approach:

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

(2) Execute a classical pivoting method on $\mathbf{\Omega}\mathbf{A}$

References: Liberty, Woolfe, Martinsson, Rokhlin and Tygert (2007); Sorensen & Embree (2006); Halko, Martinsson, Tropp (2011); Martinsson, Tropp (2020); 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**

"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.

(3) **Samples of current research directions** (time permitting)

Finding spanning row and columns.

Randomized compression of rank structured matrices.

(Matrix approximation via sampling.)

Approximation of rank-structured matrices

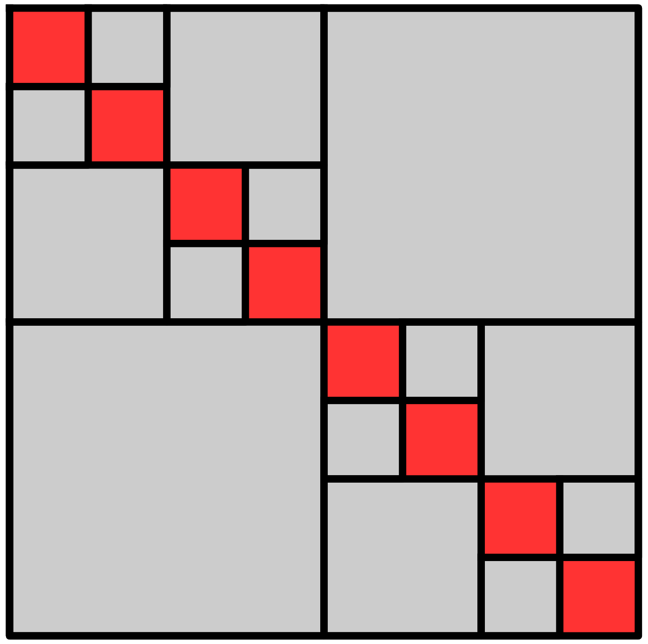
Many matrices in applications have *off-diagonal blocks* that are of low rank:

- Matrices approximating integral equations associated with elliptic PDEs.
  (Essentially, discretized Calderòn-Zygmund operators.)
- Scattering matrices in acoustic and electro-magnetic scattering.
- Inverses of (sparse) matrices arising upon FEM discretization of elliptic PDEs.
- Buzzwords: $\mathcal{H}$-matrices, HSS-matrices, HBS matrices, ...

Using randomized algorithms, we have developed $O(N)$-complexity methods for performing algebraic operations on dense matrices of this type. This leads to:

- Very fast LU factorization of FEM/FD stiffness matrices.
- Fast *direct* solvers for boundary integral equations in 2D and 3D.
- Linear or close to linear complexity methods attained in many environments.

A representative tessellation of a rank-structured matrix. Each off-diagonal block (gray) has low numerical rank. The diagonal blocks (red) are full rank, but are small in size. Matrices of this type allow efficient matrix-vector multiplication, matrix inversion, etc.



Approximation of rank-structured matrices – objective

Recall: In the randomized SVD, the input is a matrix $\mathbf{A}$ of *low numerical rank*, and an efficient way to evaluate

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

for thin matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$. Then $\mathbf{A}$ is reconstructed from the information in $\{\boldsymbol{\Omega},\, \mathbf{Y},\, \boldsymbol{\Psi},\, \mathbf{Z}\}$.

Approximation of rank-structured matrices – objective

Recall: In the randomized SVD, the input is a matrix $\mathbf{A}$ of *low numerical rank*, and an efficient way to evaluate

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

for thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$. Then $\mathbf{A}$ is reconstructed from the information in $\{\mathbf{\Omega}, \mathbf{Y}, \mathbf{\Psi}, \mathbf{Z}\}$.

New objective: Suppose that $\mathbf{A}$ is *rank-structured*, and that we can efficiently evaluate

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

for tall thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$. We can still reconstruct $\mathbf{A}$ from $\{\mathbf{\Omega}, \mathbf{Y}, \mathbf{\Psi}, \mathbf{Z}\}$!

Approximation of rank-structured matrices – objective

Recall: In the randomized SVD, the input is a matrix $\mathbf{A}$ of *low numerical rank*, and an efficient way to evaluate

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

for thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$. Then $\mathbf{A}$ is reconstructed from the information in $\{\mathbf{\Omega}, \mathbf{Y}, \mathbf{\Psi}, \mathbf{Z}\}$.

New objective: Suppose that $\mathbf{A}$ is *rank-structured*, and that we can efficiently evaluate

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

for tall thin matrices $\mathbf{\Omega}$ and $\mathbf{\Psi}$. We can still reconstruct $\mathbf{A}$ from $\{\mathbf{\Omega}, \mathbf{Y}, \mathbf{\Psi}, \mathbf{Z}\}$!

Applications:

- 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 – well established algorithms

Let $\mathbf{A}$ be a rank-structured matrix, for which we can rapidly evaluate $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$ and $\mathbf{Z} = \mathbf{A}^{*}\mathbf{\Psi}$.

Case 1: Suppose that in addition to matvec, we can also evaluate individual entries of $\mathbf{A}$. Then an HBS (a.k.a. HSS) representation can be computed in $O(N)$ operations.
Very computationally efficient in practice — requires only $O(k)$ matvecs.

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

Case 2: If all we have is the matvec, then we can still compute a rank-structured representation of $\mathbf{A}$ using so called "peeling" algorithms. The price we have to pay is that we now need $O(k \times \log N)$ matvecs involving $\mathbf{A}$ and $\mathbf{A}^{*}$.

The method is still fast in many situations, and does save messy coding work. For instance, without this black-box method, implementing the matrix-matrix multiplication, or changing the partition tree, are quite hard to implement efficiently.

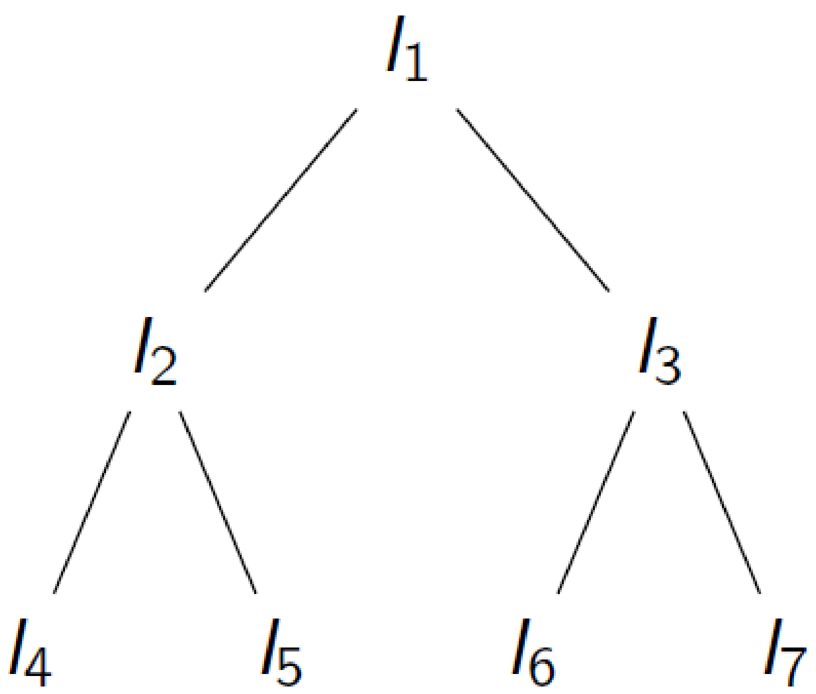
- L. Lin, J. Lu, L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, JCP 2011.
- P.G. Martinsson, SISC, **38**(4), pp. A1959-A1986, 2016.

Approximation of rank-structured matrices

Next, very recent work. A method that is both “black box” and attains true $O(N)$ complexity. (Or so I hope, at least.)

Approximation of rank-structured matrices – A binary tree structure

An example binary tree structure for a matrix of size 400×400. The levels of the tree represent successively refined partitions of the index vector $[1, ..., 400]$.



Level 0: $I_1 = [1, 2, ..., 400]$

Level 1:
$I_2 = [1, ..., 200]$, $I_3 = [201, ..., 400]$

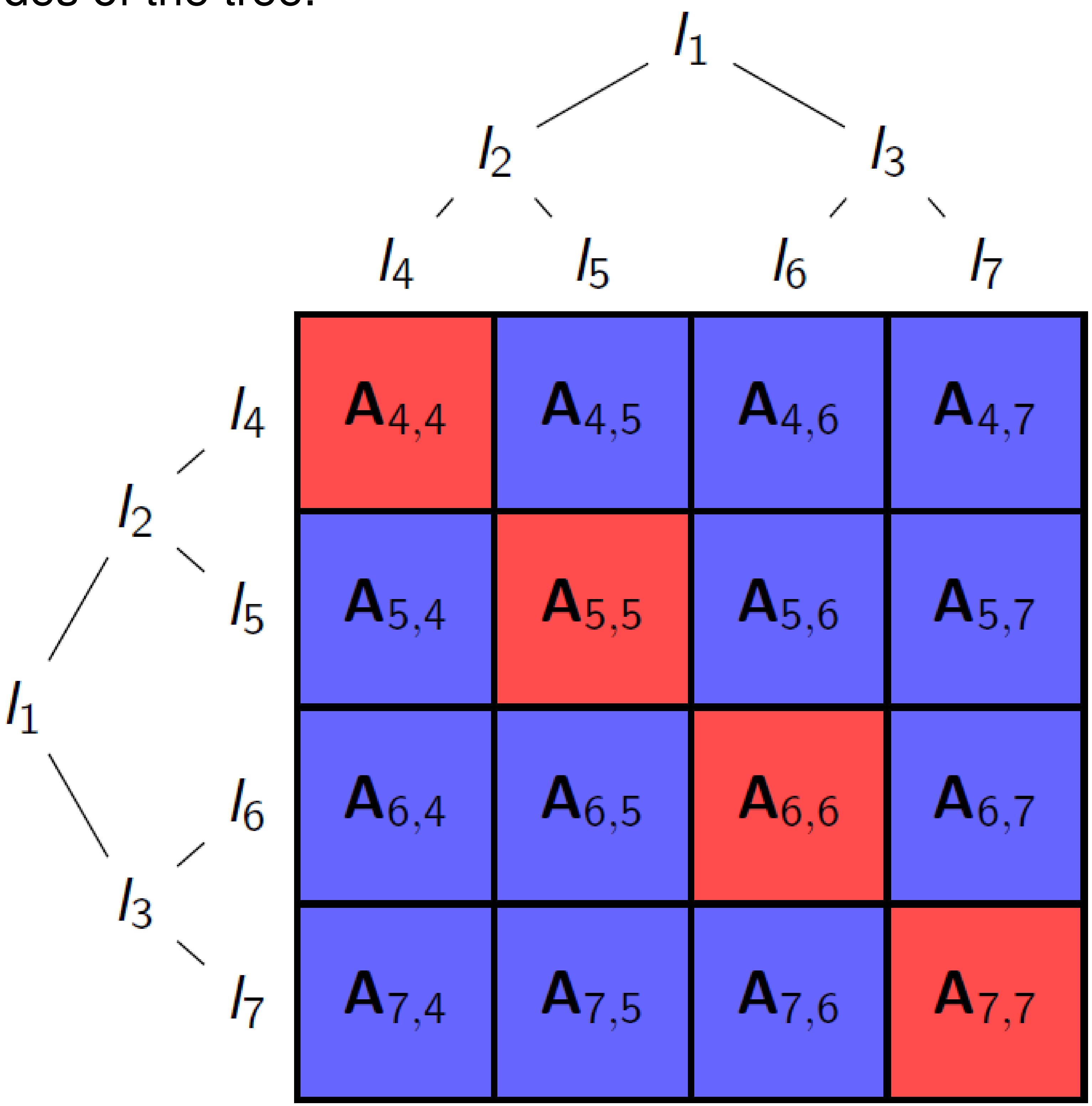
Level 2:
$I_4 = [1, ..., 100]$, $I_5 = [101, ..., 200]$,
$I_6 = [201, ..., 300]$, $I_7 = [301, ..., 400]$

Let m denote the leaf node size.

Let $L \approx \log(N/m)$ denote the depth of the tree.

Approximation of rank-structured matrices – HBS (a.k.a. HSS) structure

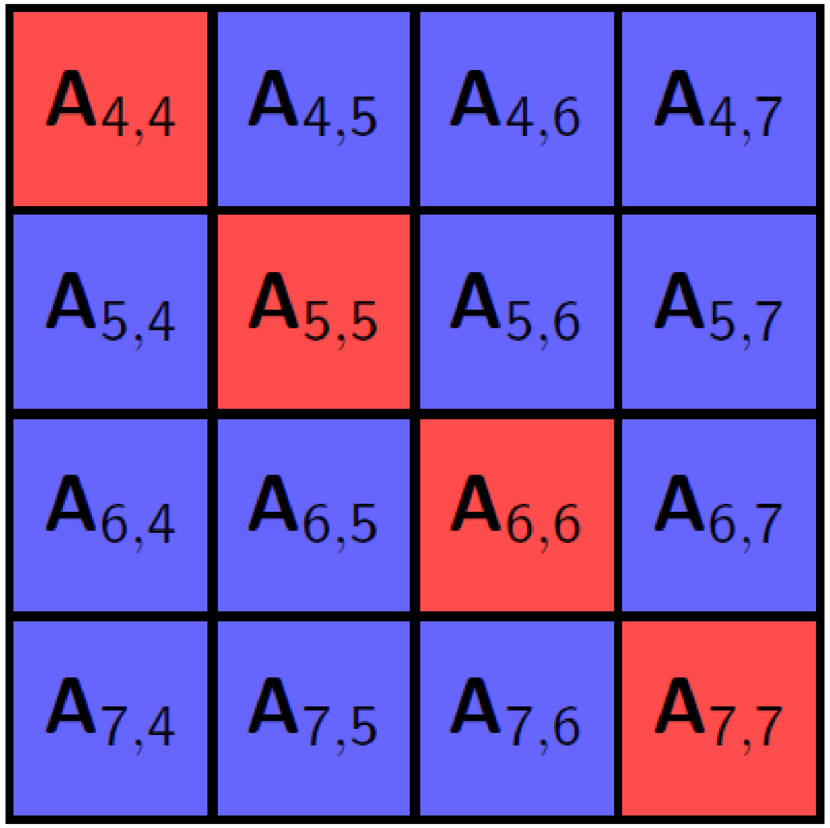
Consider the following tessellation of a matrix, where each block represents interactions between two leaf nodes of the tree.



Approximation of rank-structured matrices – HBS (a.k.a. HSS) structure

HBS requirements for the finest level: for every leaf node τ, there must exist basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ such that for every leaf node $\tau' \neq \tau$, we have

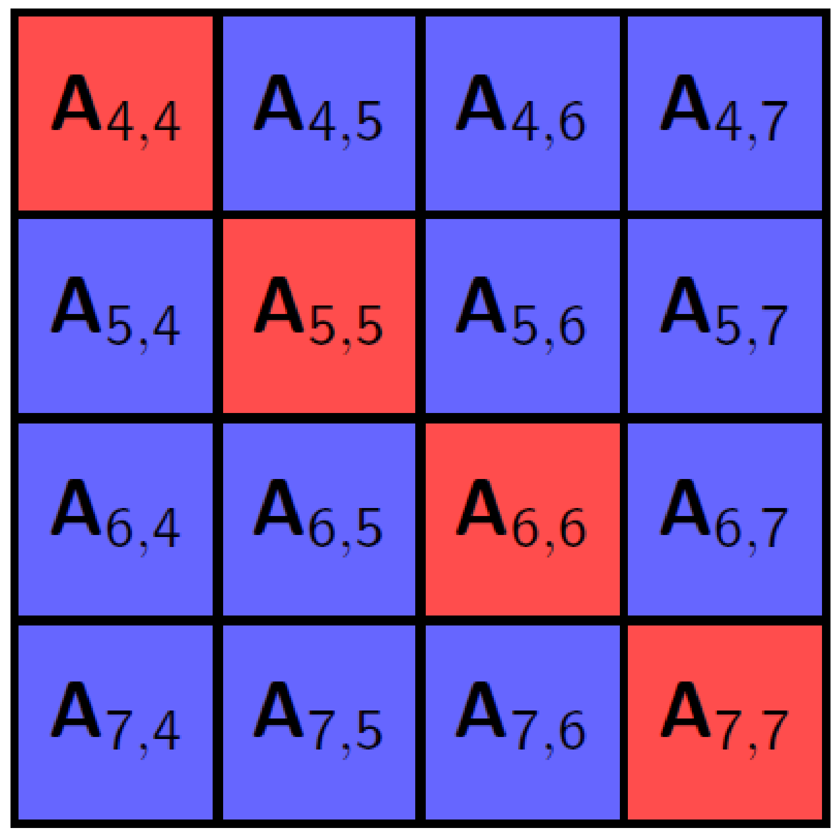
$$\begin{array}{ccccc} \mathbf{A}_{\tau,\tau'} & = & \mathbf{U}_\tau & \tilde{\mathbf{A}}_{\tau,\tau'} & \mathbf{V}^*_{\tau'}. \\ m \times m & & m \times k & k \times k & k \times m \end{array}$$



Approximation of rank-structured matrices – HBS (a.k.a. HSS) structure

HBS requirements for the finest level: for every leaf node τ, there must exist basis matrices $\mathbf{U}_\tau$ and $\mathbf{V}_\tau$ such that for every leaf node $\tau' \neq \tau$, we have

$$\underset{m\times m}{\mathbf{A}_{\tau,\tau'}} \;=\; \underset{m\times k}{\mathbf{U}_\tau}\;\underset{k\times k}{\tilde{\mathbf{A}}_{\tau,\tau'}}\;\underset{k\times m}{\mathbf{V}^*_{\tau'}}.$$

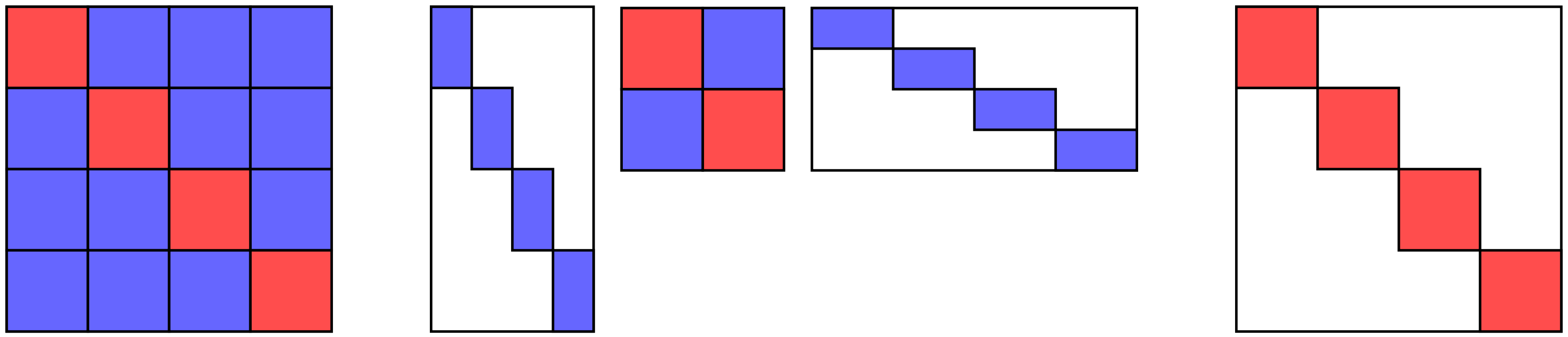


The on-diagonal blocks are not assumed to be low-rank, and pose the main challenge for black-box compression.

Approximation of rank-structured matrices – Telescoping factorization

This leads to a factorization of $\mathbf{A}$.

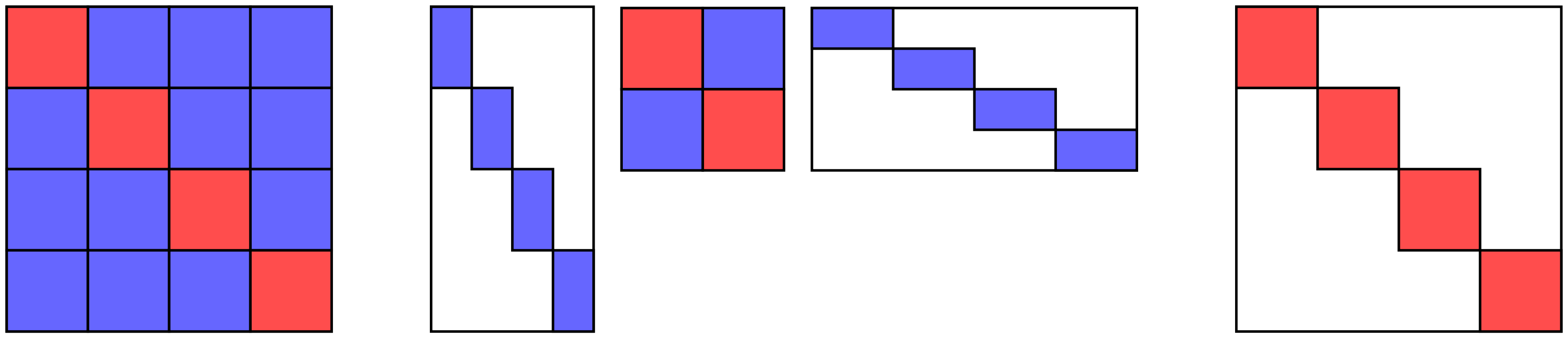
$$\mathbf{A} = \mathbf{U}^{(L)} \quad \tilde{\mathbf{A}}^{(L)} \quad (\mathbf{V}^{(L)})^{*} + \mathbf{D}^{(L)}$$



Approximation of rank-structured matrices – Telescoping factorization

This leads to a factorization of **A**.

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

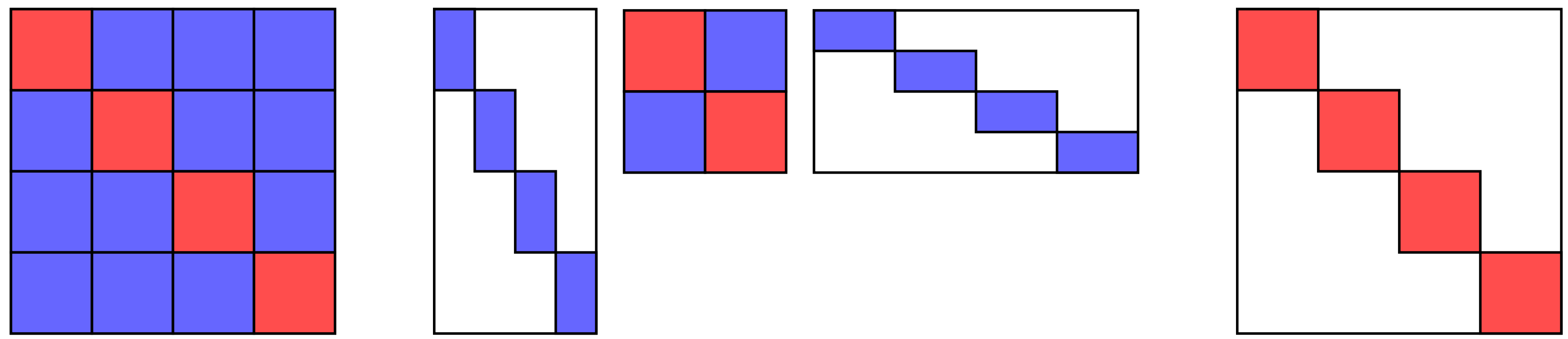


$\tilde{\mathbf{A}}^{(L)}$ is also an HBS matrix, and it can be factorized similarly, leading to a *telescoping factorization*.

Approximation of rank-structured matrices – Telescoping factorization

This leads to a factorization of $\mathbf{A}$.

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



$\tilde{\mathbf{A}}^{(L)}$ is also an HBS matrix, and it can be factorized similarly, leading to a *telescoping factorization*.

For example, a factorization of an HBS matrix with a tree of depth $L = 3$ takes the form

$$\mathbf{A} = \mathbf{U}^{(3)}(\mathbf{U}^{(2)}(\mathbf{U}^{(1)}\mathbf{D}^{(0)}(\mathbf{V}^{(1)})^* + \mathbf{D}^{(1)})(\mathbf{V}^{(2)})^* + \mathbf{D}^{(2)})(\mathbf{V}^{(3)})^* + \mathbf{D}^{(3)}.$$

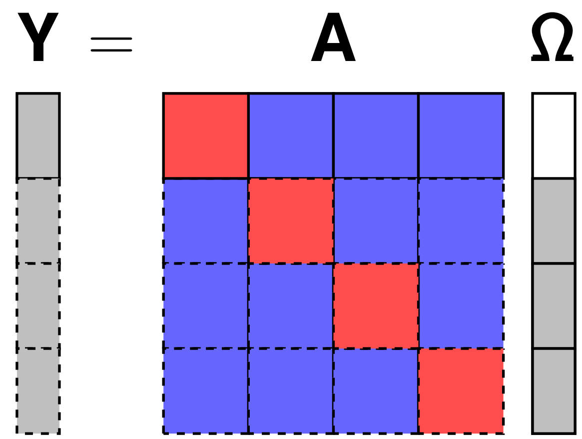
Approximation of rank-structured matrices – A naive approach

Consider the task of finding 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 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.

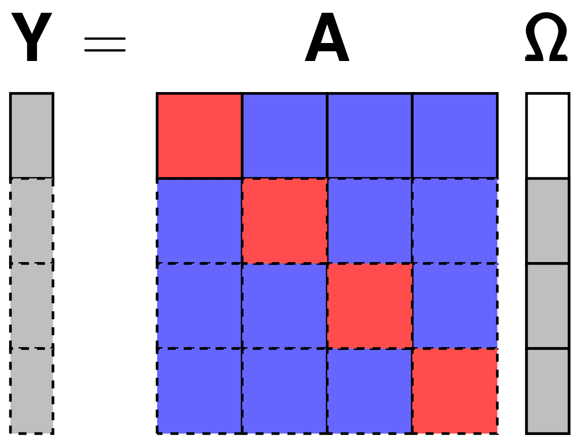
The naive approach is to sample with a random matrix $\mathbf{\Omega} \in \mathbb{R}^{N \times r}$, $r = k + 10$, that has a block of zeros in rows indexed by I_4. Then $\mathbf{Y}(I_4, :)$ will contain a sample of $\mathbf{A}(I_4, I_4^c)$.



Approximation of rank-structured matrices – A naive approach

Consider the task of finding 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.

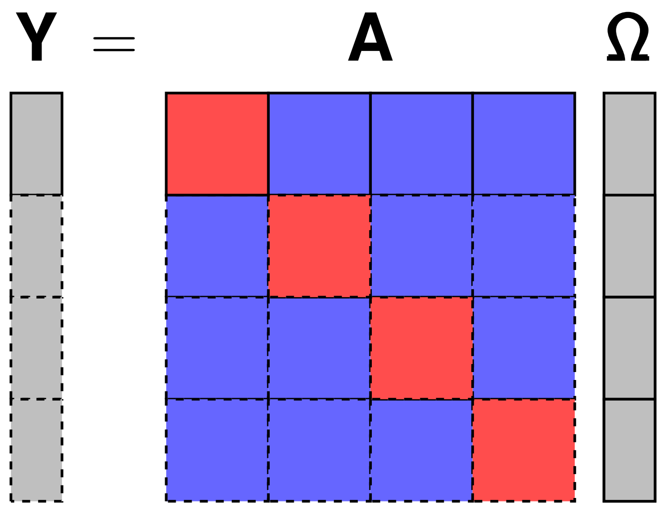
The naive approach is to sample with a random matrix $\mathbf{\Omega} \in \mathbb{R}^{N\times r}$, $r = k + 10$, that has a block of zeros in rows indexed by I_4. Then $\mathbf{Y}(I_4, :)$ will contain a sample of $\mathbf{A}(I_4, I_4^c)$.



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 "not quite black box" approach

Sample $\mathbf{A}$ with a completely dense random matrix $\mathbf{\Omega} \in \mathbb{R}^{N \times r}$.



Afterwards, subtract unwanted contributions back out of $\mathbf{Y}$.

Subtracting the contribution of $\mathbf{A}_{4,4}$ gives the desired sample of $\mathbf{A}(I_4, I_4^c)$,

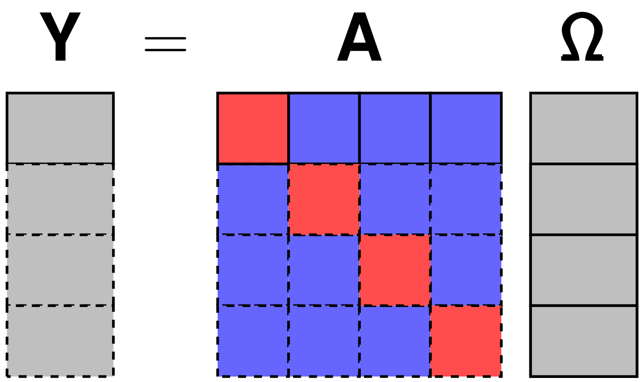
$$\mathbf{Y}(I_4, :) - \mathbf{A}_{4,4}\mathbf{\Omega}(I_4, :).$$

This scheme requires only r samples in total, but it also requires direct access to a small number of entries of $\mathbf{A}$.

P.G. Martinsson, arXiv:0806.2339, 2008. Journal version in SIMAX (Ha!), 2011

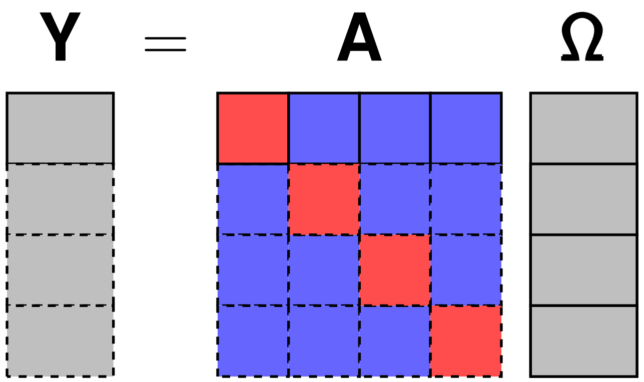
Approximation of rank-structured matrices – Finding U

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.



Approximation of rank-structured matrices – Finding U

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.



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

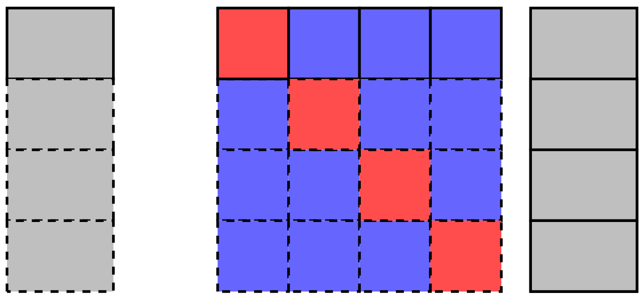
$$\mathbf{Q}_4 = \texttt{nullspace}(\mathbf{\Omega}(I_4, :), r)$$

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

Approximation of rank-structured matrices – Finding U

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

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



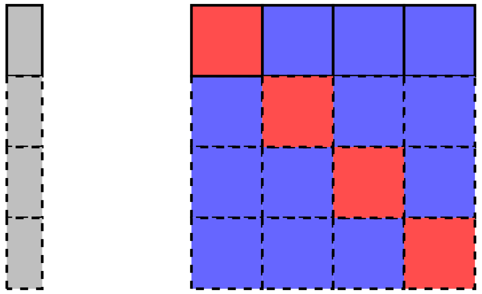
Since $\mathbf{\Omega}(I_4, :\ )$ is of size $m \times (r+m)$, it has a nullspace of dimension at least r. Let

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

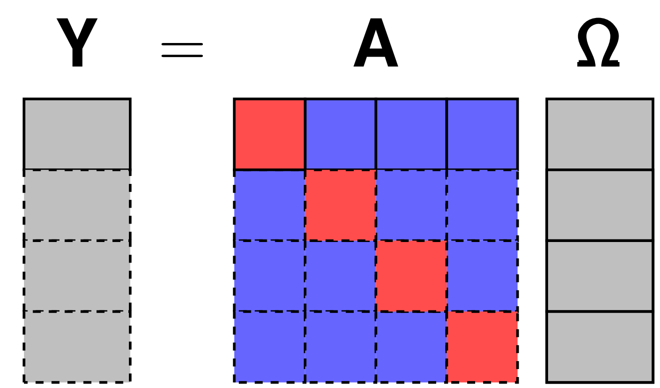
be an $(r+m) \times r$ orthonormal basis of the nullspace of $\mathbf{\Omega}(I_4, :\ )$.

Then

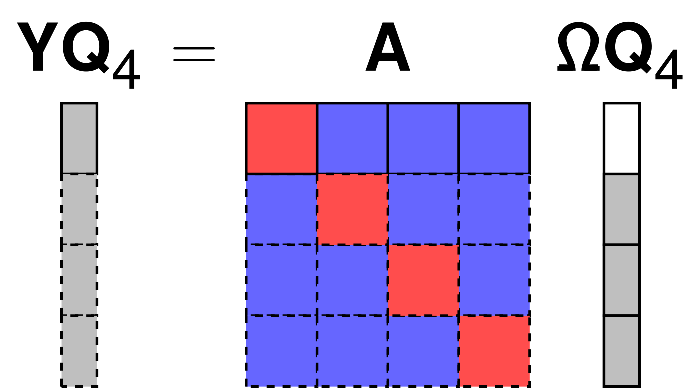
$$\mathbf{Y}\mathbf{Q}_4 \quad = \quad \mathbf{A} \quad \mathbf{\Omega}\mathbf{Q}_4.$$



Approximation of rank-structured matrices – Finding U



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



We find the sample by multiplying $\mathbf{Y}(I_4, :)\mathbf{Q}_4$.

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 – Finding U

- For each leaf node τ, we compute

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

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

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

Approximation of rank-structured matrices – Compression overview

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 $\mathbf{A}$ *and* $\mathbf{A}^*$.

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 τ in level ℓ **do**

        if τ is a leaf node **then**

$$\mathbf{\Omega}_\tau = \mathbf{\Omega}(I_\tau, :), \qquad \mathbf{\Psi}_\tau = \mathbf{\Psi}(I_\tau, :)$$
$$\mathbf{Y}_\tau = \mathbf{Y}(I_\tau, :), \qquad \mathbf{Z}_\tau = \mathbf{Z}(I_\tau, :)$$

        else

            Let α and β denote the children of τ.

$$\mathbf{\Omega}_\tau = \begin{bmatrix} \mathbf{V}_\alpha^*\mathbf{\Omega}_\alpha \\ \mathbf{V}_\beta^*\mathbf{\Omega}_\beta \end{bmatrix}, \qquad \mathbf{\Psi}_\tau = \begin{bmatrix} \mathbf{U}_\alpha^*\mathbf{\Psi}_\alpha \\ \mathbf{U}_\beta^*\mathbf{\Psi}_\beta \end{bmatrix}$$
$$\mathbf{Y}_\tau = \begin{bmatrix} \mathbf{U}_\alpha^*(\mathbf{Y}_\alpha - \mathbf{D}_\alpha\mathbf{\Omega}_\alpha) \\ \mathbf{U}_\beta^*(\mathbf{Y}_\beta - \mathbf{D}_\beta\mathbf{\Omega}_\beta) \end{bmatrix}, \quad \mathbf{Z}_\tau = \begin{bmatrix} \mathbf{V}_\alpha^*(\mathbf{Z}_\alpha - \mathbf{D}_\alpha^*\mathbf{\Psi}_\alpha) \\ \mathbf{V}_\beta^*(\mathbf{Z}_\beta - \mathbf{D}_\beta^*\mathbf{\Psi}_\beta) \end{bmatrix}$$

        if level $\ell > 0$ **then**

$$\mathbf{Q}_\tau = \mathtt{nullspace}(\mathbf{\Omega}_\tau, r), \qquad \mathbf{P}_\tau = \mathtt{nullspace}(\mathbf{\Psi}_\tau, r)$$
$$\mathbf{U}_\tau = \mathtt{qr}(\mathbf{Y}_\tau\mathbf{Q}_\tau, r), \qquad \mathbf{V}_\tau = \mathtt{qr}(\mathbf{Z}_\tau\mathbf{P}_\tau, r)$$

$$\mathbf{D}_\tau = (\mathbf{I} - \mathbf{U}_\tau\mathbf{U}_\tau^*)\mathbf{Y}_\tau\mathbf{\Omega}_\tau^\dagger + \mathbf{U}_\tau\mathbf{U}_\tau^*\left((\mathbf{I} - \mathbf{V}_\tau\mathbf{V}_\tau^*)\mathbf{Z}_\tau\mathbf{\Psi}_\tau^\dagger\right)^*$$

        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

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

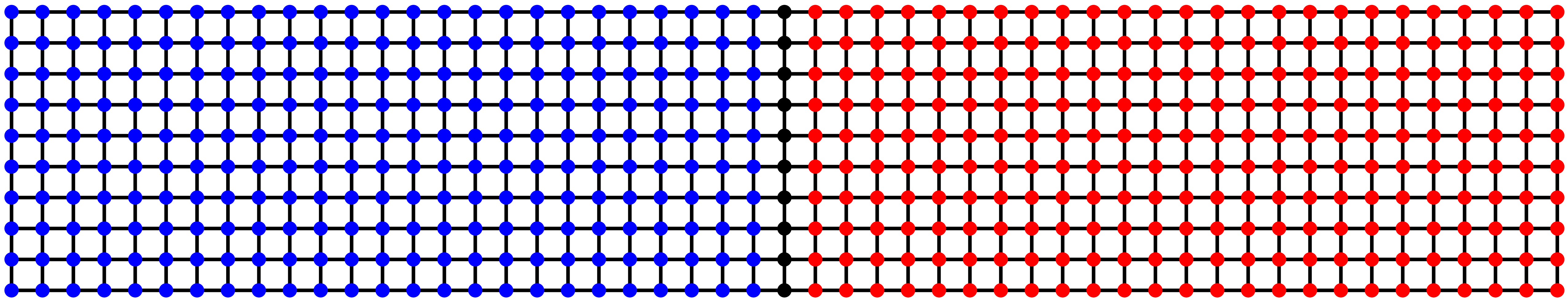
and rearrange to obtain

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

Approximation of rank-structured matrices: Sparse LU

Let $\mathbf{C}$ be the stiffness matrix for the standard five-point stencil finite difference approximation to the Poisson equation on a rectangular grid.

We partition the grid as shown and tessellate $\mathbf{C}$ accordingly.



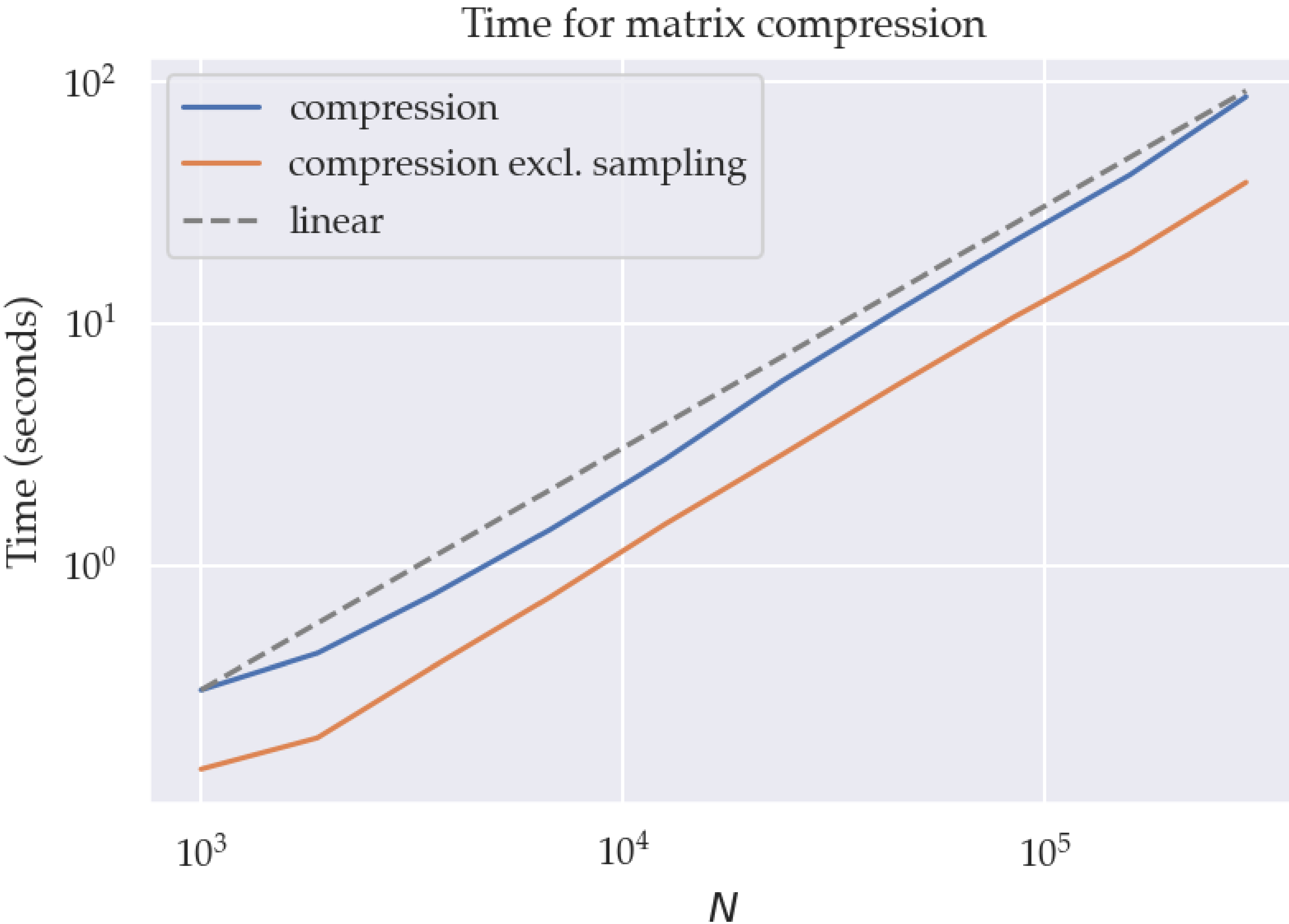
$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{0} & \mathbf{C}_{13} \\ \mathbf{0} & \mathbf{C}_{22} & \mathbf{C}_{23} \\ \mathbf{C}_{31} & \mathbf{C}_{32} & \mathbf{C}_{33} \end{bmatrix}$$

The matrix we seek to compress is the Schur complement

$$\mathbf{A} = \mathbf{C}_{33} - \mathbf{C}_{31}\mathbf{C}_{11}^{-1}\mathbf{C}_{31} - \mathbf{C}_{32}\mathbf{C}_{22}^{-1}\mathbf{C}_{23}.$$

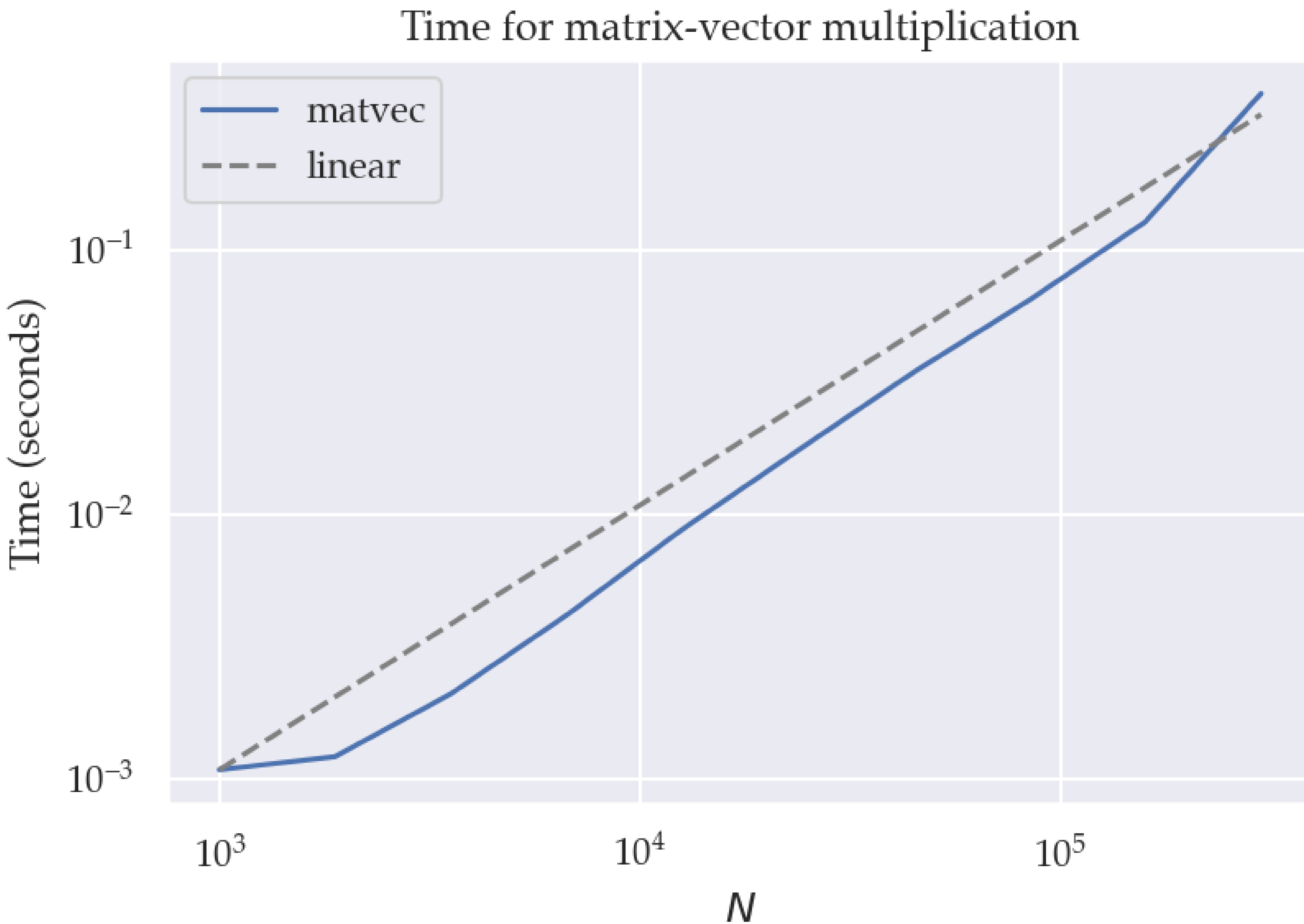
Approximation of rank-structured matrices: Sparse LU

$r = 30, m = 60$



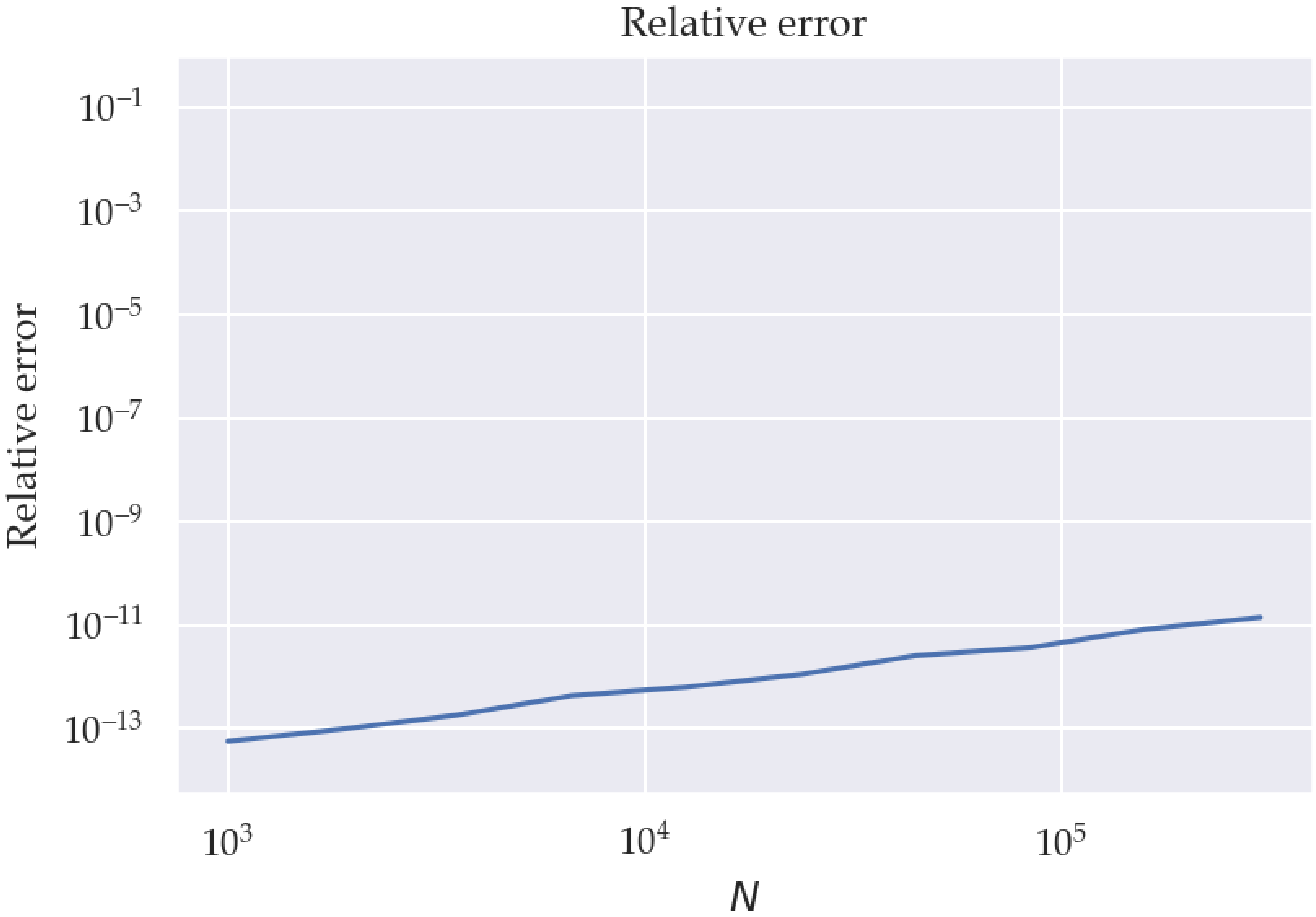
Approximation of rank-structured matrices: Sparse LU

$r = 30, m = 60$



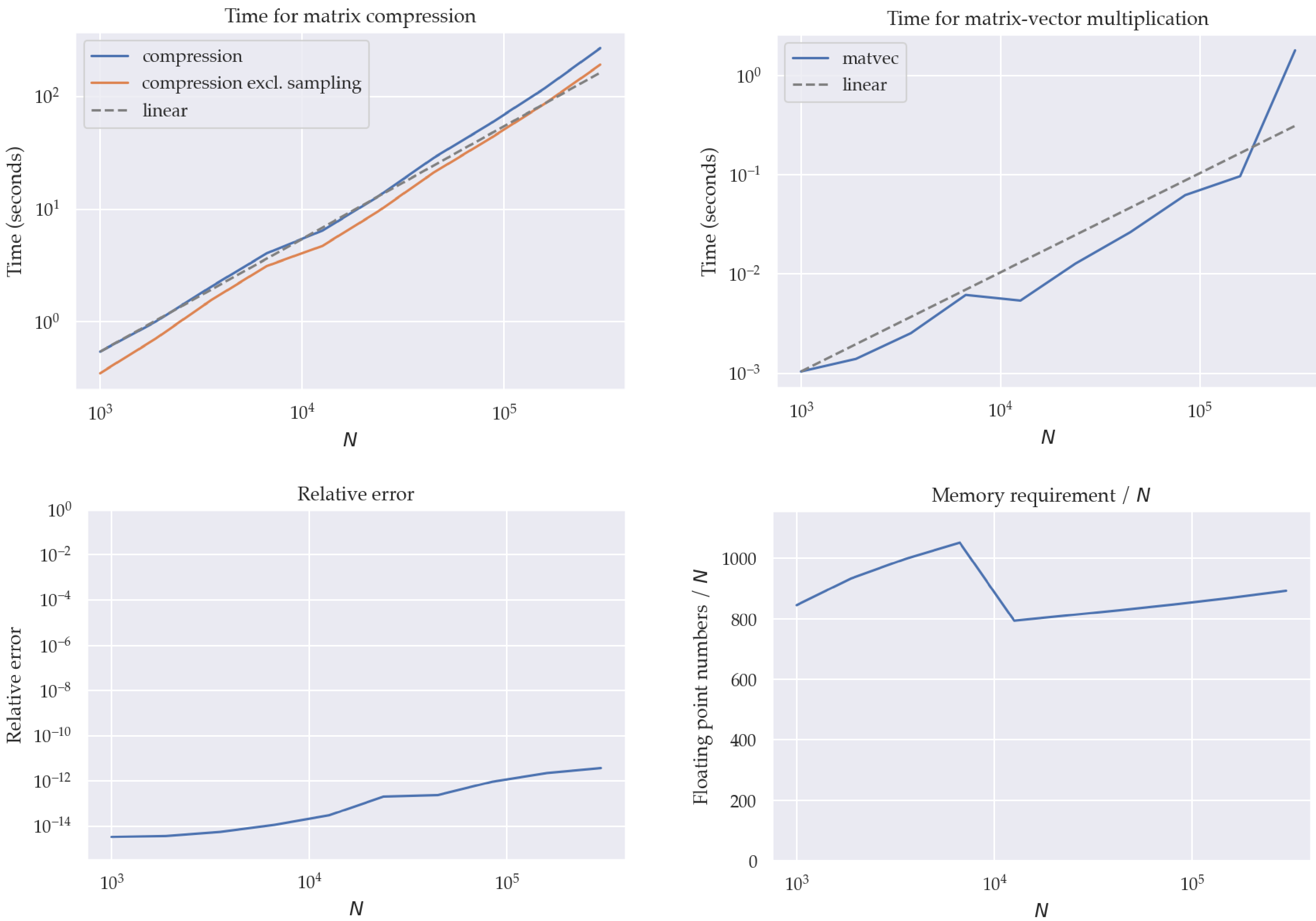
Approximation of rank-structured matrices: Sparse LU

$r = 30, m = 60$



Approximation of rank-structured matrices: FMM

$r = 50, m = 100$



Approximation of rank-structured matrices: Key points

- Fully “black box”. Interacts with $\mathbf{A}$ only via the matvec.
- True linear complexity. Requires only $O(k)$ samples from $\mathbf{A}$ and $\mathbf{A}^*$.
  Much faster in practice than existing black box algorithms.
  (However, prefactor in # samples is slightly suboptimal – unlike Townsend/Halikias.)
- Ideal tool for acceleration of sparse direct solvers.

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.

(3) **Samples of current research directions** (time permitting)

Randomized compression of rank structured matrices.

Finding spanning row and columns.

Matrix approximation via sampling.

Matrix approximation by sampling

To simplify slightly, there are two paradigms for how to use randomization to approximate matrices:

Randomized embeddings

(What we have discussed so far.)

Randomized sampling

(What we will discuss next.)

Matrix approximation by sampling

To simplify slightly, there are two paradigms for how to use randomization to approximate matrices:

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.

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

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

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

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

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

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

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 $k \sim \frac{1}{\epsilon^2}$ 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* $\mathbf{A}$ *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* **A** *into such a matrix!* Let $\mathbf{\Omega}$ be a matrix drawn from a uniform distribution on the set of $n \times n$ unitary matrices (the "Haar distribution"). Then form

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

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

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

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

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

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

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 $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, and then form $\mathbf{A}_k = \mathbf{Y}\mathbf{Y}^{\dagger}\mathbf{A}$.

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 simply unaffordable.

  Kernel matrices in data analysis form a prime target. These are dense matrices, and you just cannot form the entire matrix.

- Popular topic for theory papers.

- When techniques based on randomized embeddings that systematically mix all coordinates *are* affordable, they perform far better. Higher accuracy, and less variability in the outcome.

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.
- Compression of continuum operators and rank structured matrices.
- 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)