Brummer and Partners MathDataLab mini course

Randomized methods in linear algebra and their applications in data science

November 17 – 19, 2020

Per-Gunnar Martinsson

The University of Texas at Austin

Slides and links at: `http://users.oden.utexas.edu/~pgm/2020_kth_course/`

Research support by:

Course objectives:

The course will provide an introduction to a set of randomized algorithms for solving large scale problems in linear algebra and data analysis. Specific topics covered include:

- *Mathematical properties of random embeddings:* Using random matrices to embed high dimensional datasets into lower dimensional spaces. Johnson-Lindenstrauss' lemma. Fast Johnson-Lindenstrauss transforms. Nearest neighbor search.
- *Low rank approximation:* Given a large matrix $\mathbf{A}$ of small numerical rank, how do you compute an approximate factorization efficiently?
  - Faster principal component analysis.
  - Enable processing of datasets that are too large for traditional methods.
  - Streaming algorithms.
- *Solving linear systems:* Randomized algorithms for solving a linear system like $\mathbf{Ax} = \mathbf{b}$. Randomized preconditioners and regression problems. Randomized Kaczmarz algorithms. Solvers for graph Laplacians. Kernel ridge regression.
- *Matrix approximation by sampling:* Techniques for solving linear systems or factorizing matrices in situation where you cannot even form the entire matrix. Necessarily less robust. Enable solution of problems that are otherwise not solvable.

Tentative outline:

1. Randomized embeddings. *Lecture 1*
2. Low rank approximation — review of classical methods. *Lecture 1*
3. Randomized low rank approximation. *Lecture 2*
4. Single pass algorithms. *Lecture 2*
5. Matrix approximation by sampling. *Lecture 2*
6. CUR and interpolative decompositions. *Lecture 3*
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$. *Lecture 3*
8. Analysis of randomized low rank approximation. *Lecture 3 (time permitting)*

Note: Some topics may need to be skipped, depending on how quickly we move through the material.

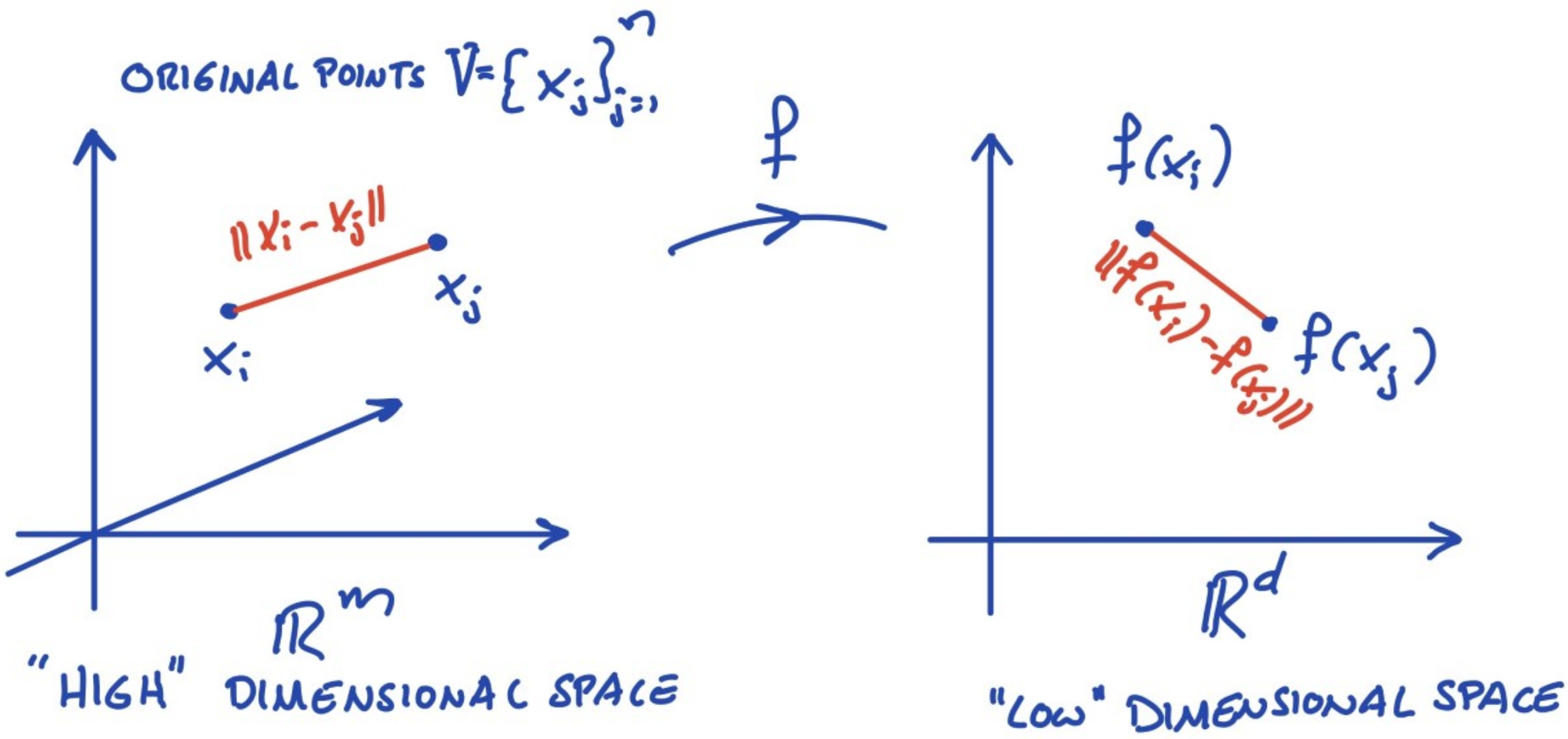
For lectures 2 and 3, feedback is welcome on topics of particular interest / indifference!

Recall: Slides and links at: `http://users.oden.utexas.edu/~pgm/2020_kth_course/`

Randomized embeddings: What are they?

Randomized embeddings: What are they?

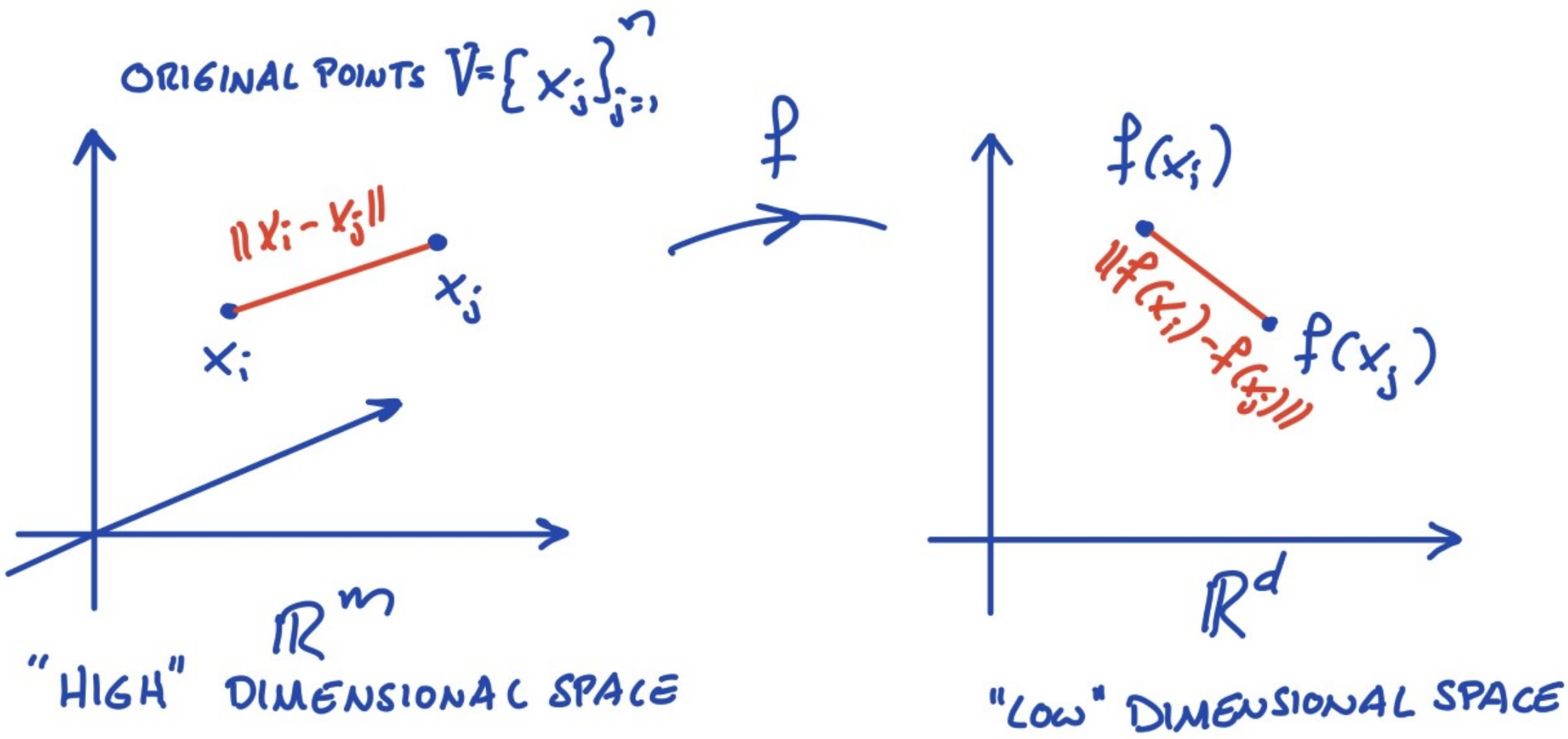
Let $V = \{\mathbf{x}_j\}_{j=1}^n$ be a set of points in $\mathbb{R}^m$, and let $f : V \to \mathbb{R}^d$ be a map, where $d < m$.
Think of m as a "large" dimension, and d as a "small" dimension.



We say that f is an *embedding* if: $\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| \approx \|\mathbf{x}_i - \mathbf{x}_j\|, \qquad \forall\, i,j \in \{1,2,\ldots,n\}.$

Randomized embeddings: What are they?

Let $V = \{\mathbf{x}_j\}_{j=1}^n$ be a set of points in $\mathbb{R}^m$, and let $f : V \to \mathbb{R}^d$ be a map, where $d < m$.
Think of m as a "large" dimension, and d as a "small" dimension.



We say that f is an *embedding* if: $\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| \approx \|\mathbf{x}_i - \mathbf{x}_j\|, \qquad \forall\, i,j \in \{1,2,\ldots,n\}.$

Lemma [Johnson-Lindenstrauss]: For $d \sim \log(n)$, there exists a well-behaved embedding f that "approximately" preserves distances.

Randomized embeddings: Applications

Consider a case where we are given a set $\{\mathbf{a}^{(j)}\}_{j=1}^{n}$ of points in $\mathbb{R}^m$, where m is very large. Now suppose that we need to solve tasks such as:

- Suppose the points almost live on a linear subspace of (small) dimension k. Find a basis for the "best" subspace. (Principal component analysis.)
- Given k, find the subset of k vectors with maximal spanning volume.
- Suppose the points almost live on a low-dimensional nonlinear manifold. Find a parameterization of the manifold.
- Given k, find for each vector $\mathbf{a}^{(j)}$ its k closest neighbors.
- Partition the points into clusters.

(Note: Some problems have well-defined solutions; some do not. The first can be solved with algorithms with moderate complexity; some are combinatorially hard.)

If we can embed the given set of points in a lower dimensional space, while approximately preserving distances, then a variety of algorithms for solving these problems become available.

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Note: This problem is very simple to solve directly!

The purpose of the randomized method we will discuss is to introduce concepts.

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Algorithm:

1. Draw a Gaussian vector $\mathbf{g} \in \mathbb{R}^n$.
2. Set $y = \mathbf{g} \cdot \mathbf{x}$ and use the estimate $\|\mathbf{x}\|^2 \approx y^2$.

Claim: The expectation of y^2 equals $\|\mathbf{x}\|^2$. (I.e. y^2 is an "unbiased estimate" for $\|\mathbf{x}\|^2$.)

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Algorithm:

1. Draw a Gaussian vector $\mathbf{g} \in \mathbb{R}^n$.
2. Set $y = \mathbf{g} \cdot \mathbf{x}$ and use the estimate $\|\mathbf{x}\|^2 \approx y^2$.

Claim: The expectation of y^2 equals $\|\mathbf{x}\|^2$. (I.e. y^2 is an "unbiased estimate" for $\|\mathbf{x}\|^2$.)

$$\mathbb{E}[y^2] = \mathbb{E}[\Big(\sum_{j=1}^{n} g_j x_j\Big)^2] = \mathbb{E}[\sum_{i,j=1}^{n} g_i g_j x_i x_j] = \sum_{i,j=1}^{n} \mathbb{E}[g_i g_j]\, x_i x_j = \sum_{i,j=1}^{n} \delta_{i,j}\, x_i x_j = \sum_{j=1}^{n} x_j^2.$$

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Algorithm:

1. Draw a Gaussian vector $\mathbf{g} \in \mathbb{R}^n$.
2. Set $y = \mathbf{g} \cdot \mathbf{x}$ and use the estimate $\|\mathbf{x}\|^2 \approx y^2$.

Claim: The expectation of y^2 equals $\|\mathbf{x}\|^2$. (I.e. y^2 is an "unbiased estimate" for $\|\mathbf{x}\|^2$.)

$$\mathbb{E}[y^2] = \mathbb{E}[\Big(\sum_{j=1}^{n} g_j x_j\Big)^2] = \mathbb{E}[\sum_{i,j=1}^{n} g_i g_j x_i x_j] = \sum_{i,j=1}^{n} \mathbb{E}[g_i g_j]\, x_i x_j = \sum_{i,j=1}^{n} \delta_{i,j}\, x_i x_j = \sum_{j=1}^{n} x_j^2.$$

Problem: y^2 is not a good estimate for $\|\mathbf{x}\|^2$, since the variance is large.

$$\begin{array}{ccccc} (\mathbf{g} \cdot \mathbf{x})^2 & = & (\cos\theta)^2 & \times & \|\mathbf{g}\|^2 \times \|\mathbf{x}\|^2 \\ & & \sim \frac{1}{n} & & \sim n \end{array}$$

Randomized embeddings: Gaussian embeddings

We will next describe how random matrices can be used to build embeddings.

Definition: We say that an $m \times n$ matrix $\mathbf{G}$ is a "Gaussian matrix" if each entry is drawn independently from a standard normal distribution. A "Gaussian vector" is defined analogously.

Warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Algorithm:

1. Draw a Gaussian vector $\mathbf{g} \in \mathbb{R}^n$.
2. Set $y = \mathbf{g} \cdot \mathbf{x}$ and use the estimate $\|\mathbf{x}\|^2 \approx y^2$.

Claim: The expectation of y^2 equals $\|\mathbf{x}\|^2$. (I.e. y^2 is an "unbiased estimate" for $\|\mathbf{x}\|^2$.)

$$\mathbb{E}[y^2] = \mathbb{E}[\Big(\sum_{j=1}^{n} g_j x_j\Big)^2] = \mathbb{E}[\sum_{i,j=1}^{n} g_i g_j x_i x_j] = \sum_{i,j=1}^{n} \mathbb{E}[g_i g_j]\, x_i x_j = \sum_{i,j=1}^{n} \delta_{i,j}\, x_i x_j = \sum_{j=1}^{n} x_j^2.$$

Problem: y^2 is not a good estimate for $\|\mathbf{x}\|^2$, since the variance is large.

Any one experiment is likely to give a substantial error.

Question: How can we improve the quality of the estimate?

Randomized embeddings: Gaussian embeddings

Recall warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Improved algorithm:

1. Pick a positive integer d. (As d grows, cost grows, and accuracy improves.)
2. Draw a $d \times n$ Gaussian matrix $\mathbf{G}$.
3. Set $\mathbf{y} = \frac{1}{\sqrt{d}}\mathbf{G}\mathbf{x}$ and use $\|\mathbf{x}\|^2 \approx \|\mathbf{y}\|^2$.

Claim: The random variable $\|\mathbf{y}\|^2$ is an "unbiased" estimate for $\|\mathbf{x}\|^2$:

Proof: An elementary computation shows that

$$\|\mathbf{y}\|^2 = \sum_{j=1}^{d} y_j^2 = \frac{1}{d}\sum_{j=1}^{d} (\mathbf{G}(j,:)\mathbf{x})^2.$$

Now observe that $\mathbf{G}(j,:)$ is a Gaussian vector, so by the proof on the previous slide, the random variable $(\mathbf{G}(j,:)\mathbf{x})^2$ has expectation $\|\mathbf{x}\|^2$.

Since $\|\mathbf{y}\|^2$ is the average of d random variables, each with expectation $\|\mathbf{x}\|^2$, its expectation is also $\|\mathbf{x}\|^2$.

Randomized embeddings: Gaussian embeddings

Recall warm up problem: Given $\mathbf{x} \in \mathbb{R}^n$, estimate its ℓ^2 norm.

Improved algorithm:

1. Pick a positive integer d. (As d grows, cost grows, and accuracy improves.)
2. Draw a $d \times n$ Gaussian matrix $\mathbf{G}$.
3. Set $\mathbf{y} = \frac{1}{\sqrt{d}}\mathbf{G}\mathbf{x}$ and use $\|\mathbf{x}\|^2 \approx \|\mathbf{y}\|^2$.

Claim: The random variable $\|\mathbf{y}\|^2$ is an "unbiased" estimate for $\|\mathbf{x}\|^2$:

Proof: An elementary computation shows that

$$\|\mathbf{y}\|^2 = \sum_{j=1}^{d} y_j^2 = \frac{1}{d}\sum_{j=1}^{d} (\mathbf{G}(j,:)\mathbf{x})^2.$$

Now observe that $\mathbf{G}(j,:)$ is a Gaussian vector, so by the proof on the previous slide, the random variable $(\mathbf{G}(j,:)\mathbf{x})^2$ has expectation $\|\mathbf{x}\|^2$.

Since $\|\mathbf{y}\|^2$ is the average of d random variables, each with expectation $\|\mathbf{x}\|^2$, its expectation is also $\|\mathbf{x}\|^2$.

Important: *The variance of $\|\mathbf{y}\|^2$ goes to zero as d grows.*

Randomized embeddings: Gaussian embeddings

We are now ready to return to our *real* question of how to embed a set $V = \{\mathbf{x}_j\}_{j=1}^n$ into a lower dimensional space.

For a positive integer d, draw a $d \times n$ Gaussian matrix $\mathbf{G}$ and set $f(\mathbf{x}) = \frac{1}{\sqrt{d}}\mathbf{G}\mathbf{x}$.

We know that $\mathbb{E}\left[\|f(\mathbf{x}) - f(\mathbf{y})\|^2\right] = \|\mathbf{x} - \mathbf{y}\|^2$, and that as d grows, the probability distribution will concentrate around the expectation.

Claim: Given an $\varepsilon > 0$, pick a positive integer

$$d \geq 4\left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3}\right)^{-1} \log(n). \tag{1}$$

Then with positive probability, we have

$$(1-\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|^2 \leq (1+\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2, \qquad \forall\, i,j \in \{1,2,\ldots,n\}. \tag{2}$$

Sketch of proof:

(1) Establish that $\frac{d}{\|\mathbf{x}\|^2}\|\mathbf{G}\mathbf{x}\|^2$ has a χ^2 distribution of degree d.

(2) Use known properties of the χ^2.

(3) Apply a simple union bound.

Randomized embeddings: Gaussian embeddings

To summarize, we have outlined a proof for:

Lemma [Johnson-Lindenstrauss]: *Let ε be a real number such that $\varepsilon \in (0, 1)$, let n be a positive integer, and let d be an integer such that*

$$d \geq 4 \left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3} \right)^{-1} \log(n). \tag{3}$$

Then for any set V of n points in $\mathbb{R}^m$, there is a map $f : \mathbb{R}^m \to \mathbb{R}^d$ such that

$$(1 - \varepsilon) \|\mathbf{u} - \mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1 + \varepsilon) \|\mathbf{u} - \mathbf{v}\|^2, \qquad \forall\, \mathbf{u}, \mathbf{v} \in V. \tag{4}$$

Randomized embeddings: Gaussian embeddings

To summarize, we have outlined a proof for:

Lemma [Johnson-Lindenstrauss]: *Let ε be a real number such that $\varepsilon \in (0, 1)$, let n be a positive integer, and let d be an integer such that*

$$d \geq 4 \left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3} \right)^{-1} \log(n). \tag{3}$$

Then for any set V of n points in $\mathbb{R}^m$, there is a map $f : \mathbb{R}^m \to \mathbb{R}^d$ such that

$$(1 - \varepsilon) \|\mathbf{u} - \mathbf{v}\|^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|^2 \leq (1 + \varepsilon) \|\mathbf{u} - \mathbf{v}\|^2, \qquad \forall\, \mathbf{u}, \mathbf{v} \in V. \tag{4}$$

Practical problem: You have two bad choices:

(1) Pick a small ε; then you get small distortions, but a huge d since $d \sim \frac{8}{\varepsilon^2} \log(n)$.

(2) Pick ε that is not close to 0; then distortions are large.

Randomized embeddings: An example of how to use them "safely"

Suppose you are given n points $\{\mathbf{a}^{(j)}\}_{j=1}^{n}$ in $\mathbb{R}^m$. The coordinate matrix is

$$\mathbf{A} = \left[\mathbf{a}^{(1)}\ \mathbf{a}^{(2)}\ \cdots\ \mathbf{a}^{(n)}\right] \in \mathbb{R}^{m\times n}.$$

How do you find the k nearest neighbors for every point?

If m is "small" (say $m \leq 10$ or so), then you have several options; you can, e.g, sort the points into a tree based on hierarchically partitioning space (a "kd-tree").

Problem: Classical techniques of this type get very expensive as m grows.

Simple idea: Use a random map to project onto low-dimensional space. This "sort of" preserves distances. Execute a fast search there.

Randomized embeddings: An example of how to use them "safely"

Suppose you are given n points $\{\mathbf{a}^{(j)}\}_{j=1}^{n}$ in $\mathbb{R}^m$. The coordinate matrix is

$$\mathbf{A} = \left[\mathbf{a}^{(1)}\ \mathbf{a}^{(2)}\ \cdots\ \mathbf{a}^{(n)}\right] \in \mathbb{R}^{m\times n}.$$

How do you find the k nearest neighbors for every point?

If m is "small" (say $m \leq 10$ or so), then you have several options; you can, e.g, sort the points into a tree based on hierarchically partitioning space (a "kd-tree").

Problem: Classical techniques of this type get very expensive as m grows.

Simple idea: Use a random map to project onto low-dimensional space. This "sort of" preserves distances. Execute a fast search there.

Improved idea: The output from a single random projection is unreliable. But, you can repeat the experiment several times, use these to generate a list of *candidates* for the nearest neighbors, and then compute exact distances to find the k closest among the candidates.

Jones, Osipov, Rokhlin, 2011

Randomized embeddings: "Fast" Johnson-Lindenstrauss transforms

So far, the only randomized embedding we have described takes the form

$$f : \mathbb{R}^n \to \mathbb{R}^d : \mathbf{x} \mapsto \frac{1}{\sqrt{d}}\mathbf{G}\mathbf{x},$$

where $\mathbf{G}$ is a matrix drawn from a Gaussian distribution. Evaluating $f(\mathbf{x})$ costs $O(nd)$.

The cost can be reduced to $O(n \log d)$ or even less, by using *structured* random maps.

- Subsampled Fourier transforms. (Or Hadamard transform / cosine transform / ...)
- Sparse random embeddings — pick matrix that consists mostly of zeros.
- Chains of random Givens' rotations.

We provide more details later.

References: Ailon & Chazelle (2006, 2010), Woolfe, Liberty, Rokhlin, Tygert (2008), Halko, Martinsson, Tropp (2011), Clarkson & Woodruff (2013), ...

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

Low rank approximation — classical methods: Basic definitions

Let **A** be a matrix of size $m \times n$.

We say that **A** has rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{E}} \; \underset{k \times n}{\mathbf{F}}.$$

Low rank approximation — classical methods: Basic definitions

Let $\mathbf{A}$ be a matrix of size $m \times n$.

We say that $\mathbf{A}$ has rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{E}} \; \underset{k \times n}{\mathbf{F}}.$$

For $\varepsilon > 0$, we say that $\mathbf{A}$ has ε-rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{E}\mathbf{F}\| \leq \varepsilon.$$

Low rank approximation — classical methods: Basic definitions

Let **A** be a matrix of size $m \times n$.

We say that **A** has rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

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

For $\varepsilon > 0$, we say that **A** has ε-rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{EF}\| \leq \varepsilon.$$

Applications of low rank approximation:

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

Low rank approximation — classical methods: Basic definitions

Let **A** be a matrix of size $m \times n$.

We say that **A** has rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\begin{array}{ccc} \mathbf{A} & = & \mathbf{E} \quad \mathbf{F}. \\ m \times n & & m \times k \;\; k \times n \end{array}$$

For $\varepsilon > 0$, we say that **A** has ε-rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{EF}\| \leq \varepsilon.$$

The fixed rank approximation problem:

Given k, find matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\mathbf{A} \approx \mathbf{EF}.$$

Low rank approximation — classical methods: Basic definitions

Let **A** be a matrix of size $m \times n$.

We say that **A** has rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

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

For $\varepsilon > 0$, we say that **A** has ε-rank k if there exist $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{EF}\| \leq \varepsilon.$$

The fixed rank approximation problem:

Given k, find matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\mathbf{A} \approx \mathbf{EF}.$$

The fixed precision approximation problem:

Given $\varepsilon > 0$, find k, and matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{EF}\| \leq \varepsilon.$$

Low rank approximation — classical methods: The SVD

Let $\mathbf{A}$ be an $m \times n$ matrix. Then $\mathbf{A}$ admits a *singular value decomposition (SVD)*

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

where $r = \min(m, n)$ and where

$\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r]$ is a matrix holding the "left singular vectors" $\mathbf{u}_i$,

$\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_r]$ is a matrix holding the "right singular vectors" $\mathbf{v}_i$,

$\mathbf{D} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ is a diagonal matrix holding the "singular values" σ_i.

For any k such that $1 \leq k \leq \min(m, n)$, we define the *truncated* SVD as

$$\mathbf{A}_k = \mathbf{U}(:, 1:k)\,\mathbf{D}(1:k, 1:k)\,\mathbf{V}(:, 1:k)^* = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^*.$$

The following theorem states that $\mathbf{A}_k$ is the "optimal" rank-k approximation to $\mathbf{A}$:

Theorem (Eckart-Young): Let $\|\cdot\|$ denote either the Frobenius or spectral norm. Then

$$\|\mathbf{A} - \mathbf{A}_k\| = \min\{\|\mathbf{A} - \mathbf{B}\| : \ \mathbf{B} \text{ is of rank } k\}.$$

Moreover,

$\|\mathbf{A} - \mathbf{A}_k\| = \sigma_{k+1}$, when the spectral norm is used,

$\|\mathbf{A} - \mathbf{A}_k\| = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_r^2}$, when the Frobenius norm is used.

Low rank approximation — classical methods: The SVD

Recall the SVD

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times r}{\mathbf{U}}\ \underset{r\times r}{\mathbf{D}}\ \underset{r\times n}{\mathbf{V}^*} = \sum_{j=1}^{r}\sigma_j\mathbf{u}_j\mathbf{v}_j^*.$$

where $r = \min(m,n)$. Some facts:

- The left singular vectors $\{\mathbf{u}_j\}_{j=1}^k$ form an optimal basis for the column space of $\mathbf{A}$ in the sense that

  $$\|\mathbf{A} - \mathbf{U}(:,1:k)\mathbf{U}(:,1:k)^*\mathbf{A}\| = \inf\{\|\mathbf{A} - \mathbf{P}\mathbf{A}\| : \text{ where } \mathbf{P} \text{ is an ON proj. to a } k-\text{dimensional space}\}.$$

- The right singular vectors $\{\mathbf{v}_j\}_{j=1}^k$ form an optimal basis for the row space of $\mathbf{A}$.

- For a *symmetric* matrix, the eigenvalue decomposition (EVD) and the singular value decomposition are in many ways equivalent, and a truncated EVD is also an optimal rank-k approximation.

- The EVD and the SVD are also in many ways equivalent for a *normal* matrix (recall that $\mathbf{A}$ is normal if $\mathbf{A}\mathbf{A}^* = \mathbf{A}^*\mathbf{A}$), but the EVD might be complex even when $\mathbf{A}$ is real.

- For *non-normal* matrices, eigenvectors and eigenvalues are generally not convenient tools for low rank approximation.

- For a general matrix, the SVD provides the EVDs of $\mathbf{A}^*\mathbf{A}$ and $\mathbf{A}\mathbf{A}^*$:

  $$\mathbf{A}\mathbf{A}^* = \mathbf{U}\mathbf{D}^2\mathbf{U}^*, \qquad \text{and} \qquad \mathbf{A}^*\mathbf{A} = \mathbf{V}\mathbf{D}^2\mathbf{V}^*.$$

Low rank approximation — classical methods: The SVD

The SVD provides answers to the low rank approximation problems:

Low rank approximation — classical methods: The SVD

The SVD provides answers to the low rank approximation problems:

The fixed rank approximation problem:

Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and k, find matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\mathbf{A} \approx \mathbf{EF}.$$

```
[U,D,V] = svd(A);
E = U(:,1:k);
F = D(1:k,1:k) V(:,1:k)';
```

Low rank approximation — classical methods: The SVD

The SVD provides answers to the low rank approximation problems:

The fixed rank approximation problem:

Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and k, find matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\mathbf{A} \approx \mathbf{EF}.$$

```
[U,D,V] = svd(A);
E = U(:,1:k);
F = D(1:k,1:k) V(:,1:k)';
```

The fixed precision approximation problem:

Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\varepsilon > 0$, find k, and matrices $\mathbf{E} \in \mathbb{R}^{m \times k}$ and $\mathbf{F} \in \mathbb{R}^{k \times n}$ such that

$$\|\mathbf{A} - \mathbf{EF}\| \leq \varepsilon.$$

```
[U,D,V] = svd(A);
k = sum(diag(D) > epsilon);
E = U(:,1:k);
F = D(1:k,1:k) V(:,1:k)';
```

This works well provided that the matrix is “small” (say $m, n \leq 5000$) and that you have access to a function for computing the SVD. Writing one from scratch is not so easy.

Low rank approximation — classical methods: Gram-Schmidt

Let **A** be an $m \times n$ matrix of low numerical rank.

Suppose that you cannot afford to compute the full SVD.

Or that you do not have such a function implemented. (It is *hard* to write. . .)

Question: How do you compute a low rank approximation to **A**?

Low rank approximation — classical methods: Gram-Schmidt

The perhaps computationally simplest method is to perform Gram-Schmidt on the columns/rows of **A**.

Given an $m \times n$ matrix **A** and a tolerance ε, find a low rank approximation of **A** that is accurate to precision ε:

(1) **for** $k = 1, 2, 3, \ldots$

(2) Let i denote the index of the largest column of **A**.

(3) Set $\mathbf{q}_k = \dfrac{\mathbf{A}(:, i)}{\|\mathbf{A}(:, i)\|}$.

(4) $\mathbf{A} = \mathbf{A} - \mathbf{q}_k \, (\mathbf{q}_k^* \mathbf{A})$

(5) **if** $\|\mathbf{A}\| \leq \varepsilon$ **then break**

(6) **end for**

(7) $\mathbf{Q} = \big[\mathbf{q}_1 \; \mathbf{q}_2 \; \ldots \mathbf{q}_k\big]$.

Once the process concludes, you know that $\|\mathbf{A} - \mathbf{Q} \, (\mathbf{Q}^* \mathbf{A})\| \leq \varepsilon$. Complexity is $O(mnk)$.

Note: *In numerical analysis, the Gram-Schmidt process is often interpreted as computing a "column pivoted QR factorization". For efficiency and numerical stability, the computation is not really organized as shown above.*

Low rank approximation — classical methods: Gram-Schmidt

Let **A** be an $m \times n$ matrix of low numerical rank.

Suppose the matrix is huge, say $m, n \sim 10^9$, but sparse.

Question: How do you compute a low rank approximation to **A**?

Low rank approximation — classical methods: Krylov methods

Pick a starting vector $\mathbf{r}$ (often a random vector), "restrict" the matrix $\mathbf{A}$ to the k-dimensionsal "Krylov subspace"

$$\text{Span}(\mathbf{r},\ \mathbf{A}\,\mathbf{r},\ \mathbf{A}^2\,\mathbf{r},\ \ldots,\ \mathbf{A}^{k-1}\,\mathbf{r})$$

and compute an eigendecomposition of the resulting matrix.

Advantages:

- Very simple access to $\mathbf{A}$.
- Extremely high accuracy possible. (Double precision accuracy for "converged" eigenmodes, etc.)
- Efficient whenever $\mathbf{A}$ can rapidly be applied to a vector. $\mathbf{A}$ could be sparse, sparse in Fourier space, amenable to "fast summation methods", etc.

Drawbacks:

- The matrix is typically revisited $O(k)$ times if a rank-k approximation is sought. (Blocked versions exist, but the convergence analysis is less developed.)
- Numerical stability issues. These are well-studied and can be overcome, but they make software less portable (between applications, hardware platforms, etc.).

Low rank approximation — classical methods: A red thread

Observation: Standard methods for computing a low rank approximation to an $m \times n$ matrix $\mathbf{A}$ result in a factorization of the form

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

where the columns of $\mathbf{Q}$ form an approximate orthonormal basis for the range of $\mathbf{A}$.

- When we compute the SVD, $\mathbf{Q}$ holds the first k left singular vectors.
- When Gram-Schmidt is used, the result is $\mathbf{A} \approx \mathbf{Q}\,\mathbf{R}\mathbf{P}^*$ where $\mathbf{R}$ is upper triangular and $\mathbf{P}$ is a permutation matrix.
- In a Krylov method, the columns of $\mathbf{Q}$ form an ON basis for $\text{Span}(\mathbf{r}, \mathbf{A}\,\mathbf{r}, \mathbf{A}^2\,\mathbf{r}, \ldots, \mathbf{A}^{k-1}\,\mathbf{r})$.

Our computational primitive: Find ON matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\,\mathbf{Q}^*\mathbf{A}$.

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$,

find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^* \mathbf{A}$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw Gaussian random vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in \mathbb{R}^n$.
2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\mathbf{g}_1, \mathbf{y}_2 = \mathbf{A}\mathbf{g}_2, \ldots, \mathbf{y}_k = \mathbf{A}\mathbf{g}_k \in \mathbb{R}^m$.
3. Form orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^m$ such that

$$\text{Span}(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k) = \text{Span}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k).$$

For instance, Gram-Schmidt can be used — pivoting is rarely required.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw Gaussian random vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in \mathbb{R}^n$.
2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\mathbf{g}_1, \mathbf{y}_2 = \mathbf{A}\mathbf{g}_2, \ldots, \mathbf{y}_k = \mathbf{A}\mathbf{g}_k \in \mathbb{R}^m$.
3. Form orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^m$ such that

$$\text{Span}(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k) = \text{Span}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k).$$

For instance, Gram-Schmidt can be used — pivoting is rarely required.

If $\mathbf{A}$ has *exact* rank k, then $\text{Span}\{\mathbf{q}_j\}_{j=1}^k = \text{Ran}(\mathbf{A})$ with probability 1.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw Gaussian random vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in \mathbb{R}^n$.
2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\mathbf{g}_1, \mathbf{y}_2 = \mathbf{A}\mathbf{g}_2, \ldots, \mathbf{y}_k = \mathbf{A}\mathbf{g}_k \in \mathbb{R}^m$.
3. Form orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^m$ such that

$$\text{Span}(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k) = \text{Span}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k).$$

For instance, Gram-Schmidt can be used — pivoting is rarely required.

If $\mathbf{A}$ has *exact* rank k, then $\text{Span}\{\mathbf{q}_j\}_{j=1}^k = \text{Ran}(\mathbf{A})$ with probability 1.

What is perhaps surprising is that even in the general case, $\{\mathbf{q}_j\}_{j=1}^k$ often does almost as good of a job as the theoretically optimal vectors.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw Gaussian random vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in \mathbb{R}^n$.
2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\mathbf{g}_1, \mathbf{y}_2 = \mathbf{A}\mathbf{g}_2, \ldots, \mathbf{y}_k = \mathbf{A}\mathbf{g}_k \in \mathbb{R}^m$.
3. Form orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k \in \mathbb{R}^m$ such that

$$\text{Span}(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k) = \text{Span}(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k).$$

For instance, Gram-Schmidt can be used — pivoting is rarely required.

If $\mathbf{A}$ has *exact* rank k, then $\text{Span}\{\mathbf{q}_j\}_{j=1}^k = \text{Ran}(\mathbf{A})$ with probability 1.

What is perhaps surprising is that even in the general case, $\{\mathbf{q}_j\}_{j=1}^k$ often does almost as good of a job as the theoretically optimal vectors.

Next, let us simply turn the vector operations into matrix operations.

Randomized low rank approximation

Range finding problem: Given an $m \times n$ matrix $\mathbf{A}$ and an integer $k < \min(m, n)$, find an orthonormal $m \times k$ matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{A}$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw a Gaussian random matrix $\mathbf{G} \in \mathbb{R}^{n \times k}$.
2. Form a "sample" matrix $\mathbf{Y} = \mathbf{A}\mathbf{G} \in \mathbb{R}^{m \times k}$.
3. Form an orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{m \times k}$ such that $\mathbf{Y} = \mathbf{Q}\mathbf{R}$.
   For instance, Gram-Schmidt can be used — pivoting is rarely required.

Randomized low rank approximation: The randomized SVD (RSVD)

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$.
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\mathbf{R}$.
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$.
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$.
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Randomized low rank approximation: The randomized SVD (RSVD)

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\mathbf{R}$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Randomized low rank approximation: The randomized SVD (RSVD)

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\mathbf{R}$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Observation: The proposed method interacts with $\mathbf{A}$ exactly twice:

- The matrix-matrix multiplication on line 2: $\mathbf{Y} = \mathbf{A}\mathbf{G}$.
- The matrix-matrix multiplication on line 4: $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$.

The matrix-matrix multiplication is a very efficient operation. It executes well on many different computing platforms — singlecore CPU, multicore CPU, distributed memory parallel machines, cloud computers. *Very* fast on GPUs.

Later, we will demonstrate that one can actually avoid the second visit to $\mathbf{A}$.
This allows us to process matrices so large they cannot be stored at all.

Randomized low rank approximation: The randomized SVD (RSVD)

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

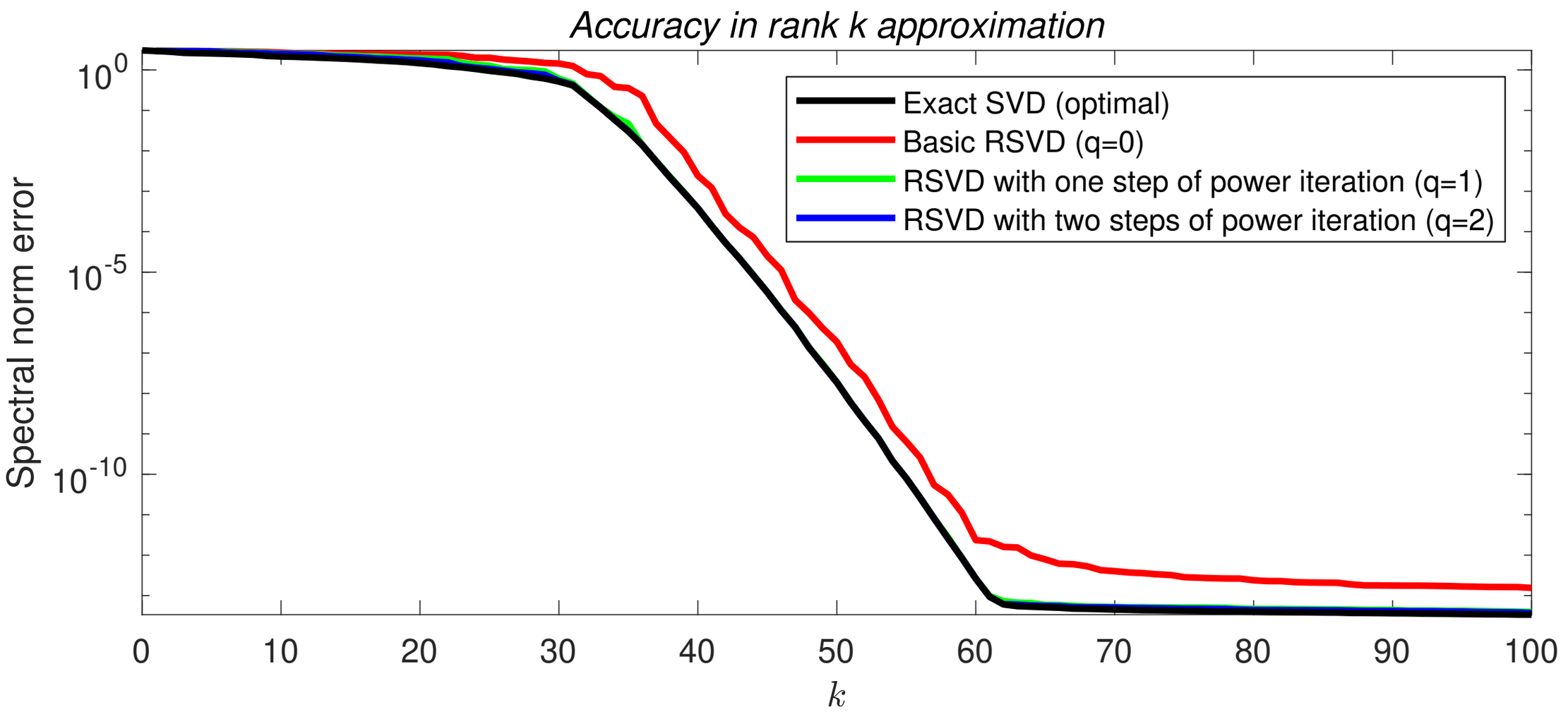
Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\mathbf{R}$. `[Q, ~] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. `U = Q * Uhat`

Power iteration to improve the accuracy: The computed factorization is close to optimally accurate when the singular values of $\mathbf{A}$ decay rapidly.

When they do not, a small amount of power iteration should be incorporated:
Replace Step 2 by $\mathbf{Y} = \left(\mathbf{A}\mathbf{A}^*\right)^t\mathbf{A}\mathbf{G}$ for a small integer t, say $t = 1$ or $t = 2$.
(Compute $\mathbf{Y}$ via alternated application of $\mathbf{A}$ and $\mathbf{A}^*$.)

Randomized low rank approximation:



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

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

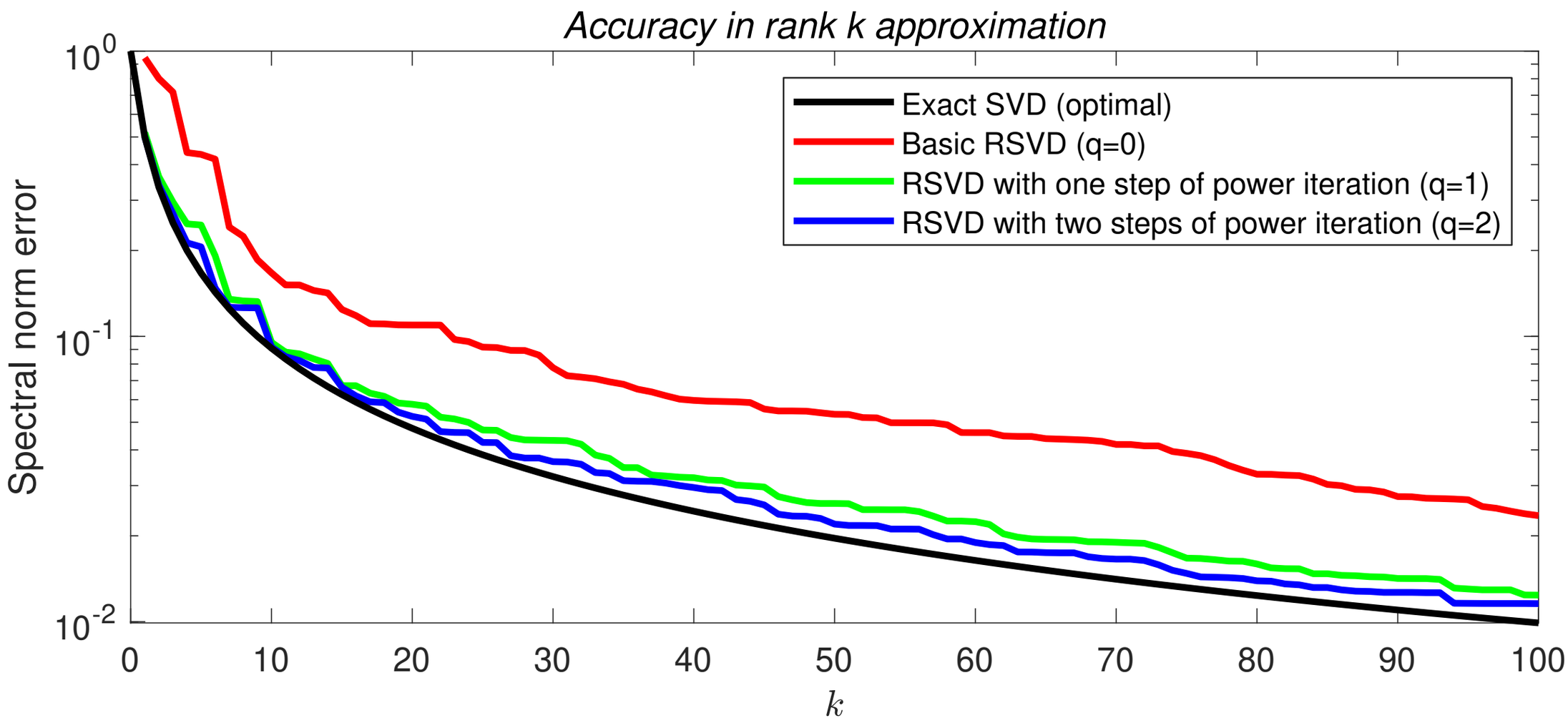
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

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

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

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

Randomized low rank approximation:



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

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

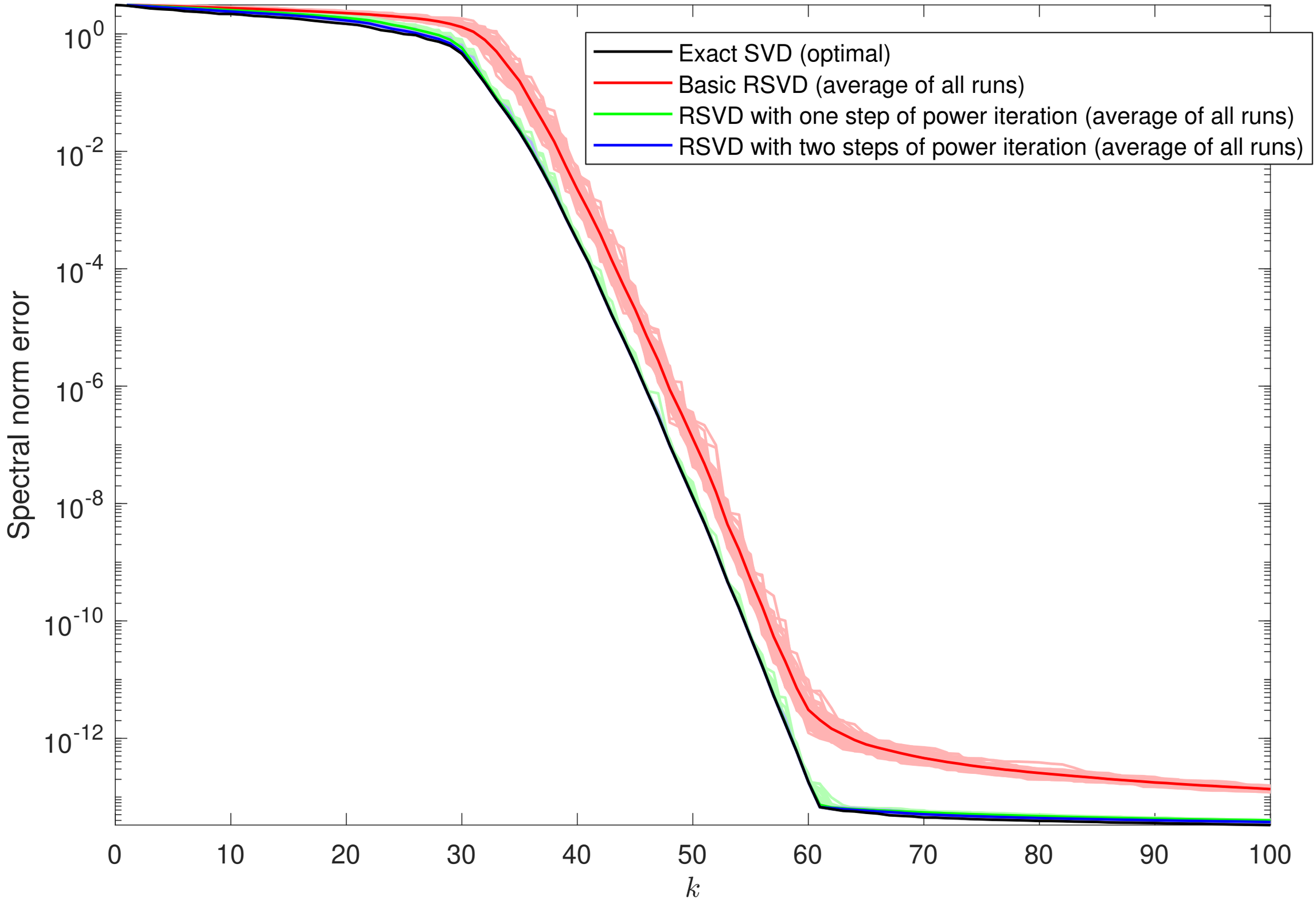
where $\mathbf{P}_k$ is the orthogonal projection onto the first k columns of

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

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

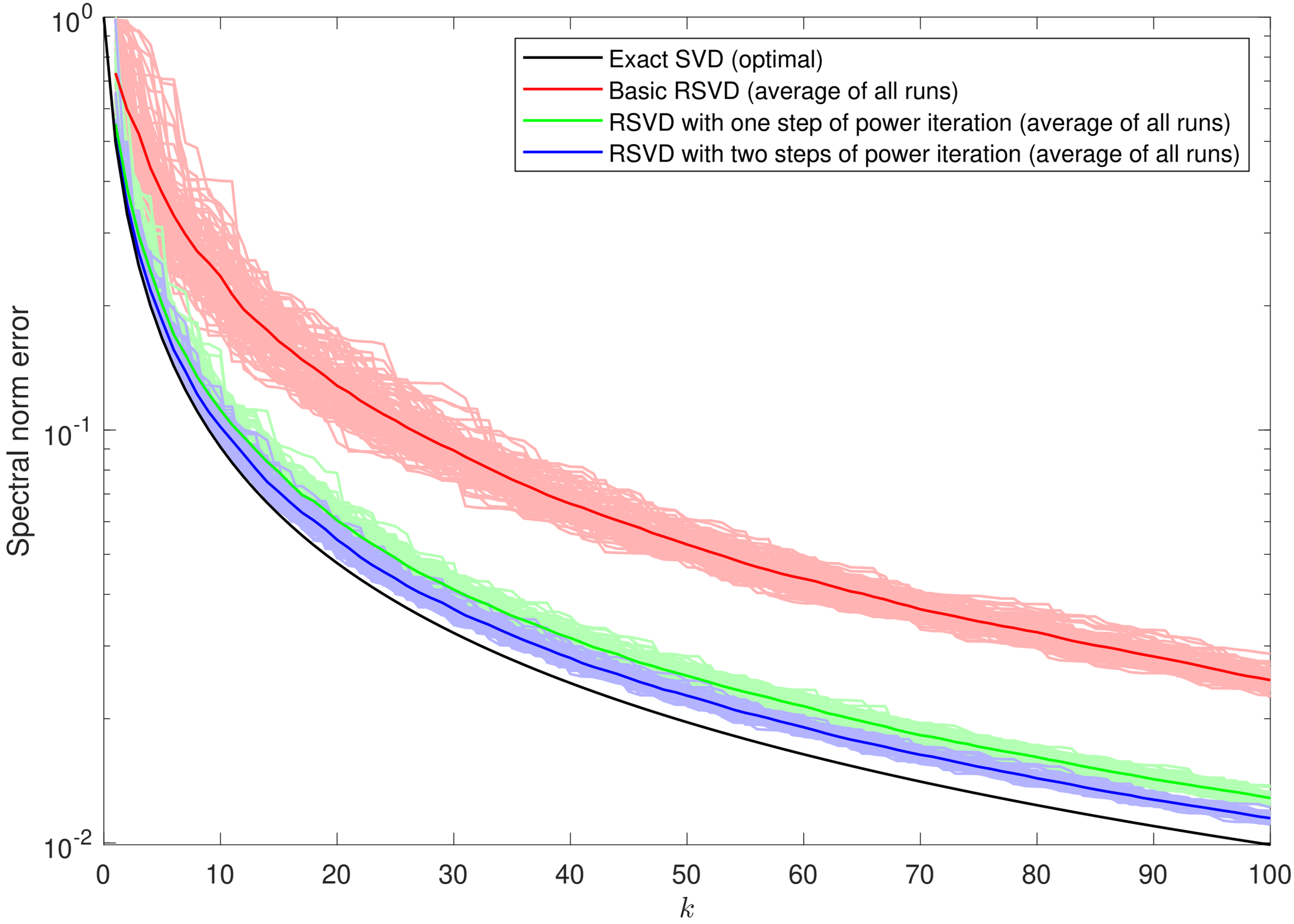
The matrix $\mathbf{A}$ now has singular values that decay slowly.

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



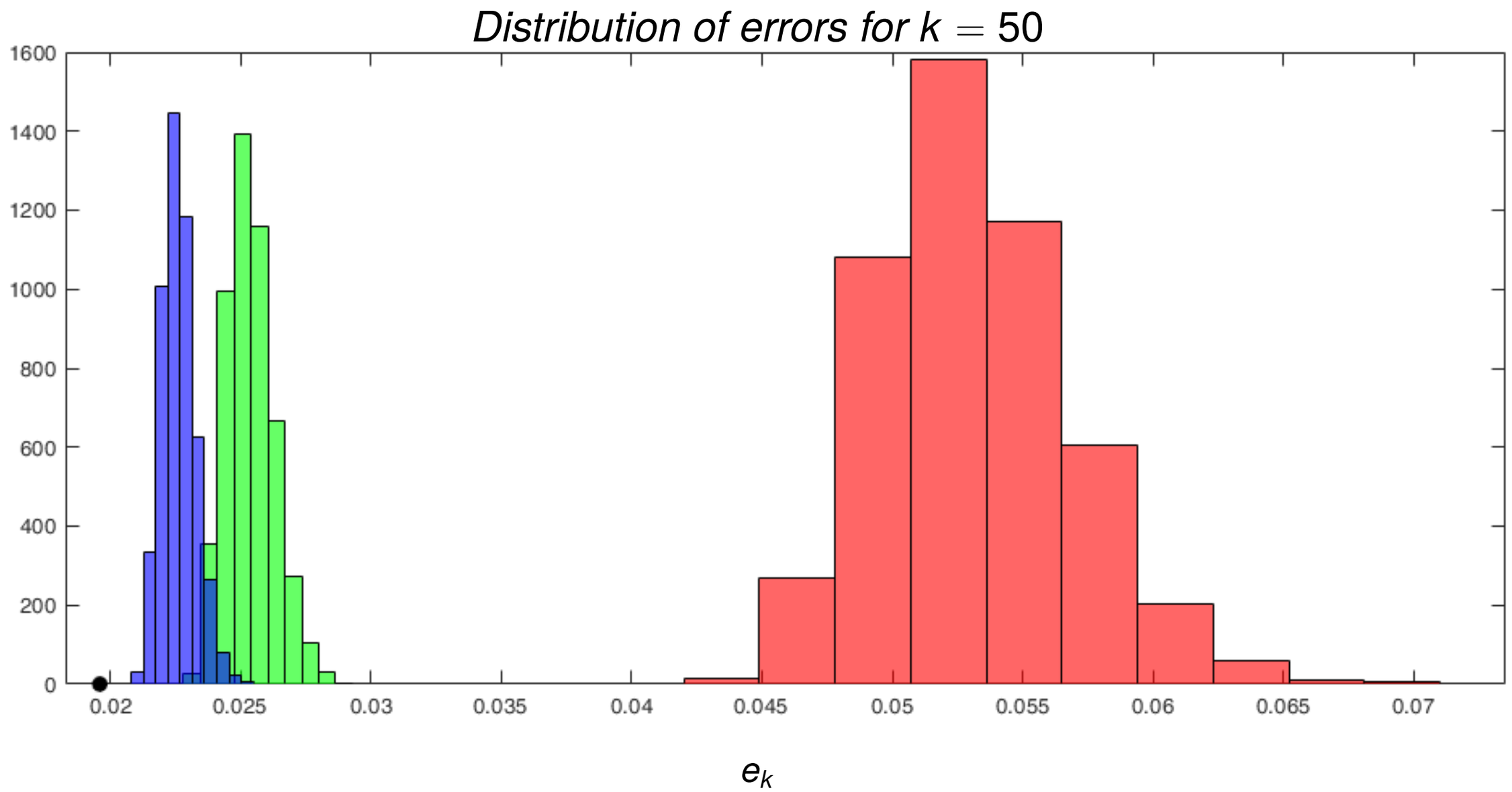
The darker lines show the mean errors across the 100 experiments.

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



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

Randomized low rank approximation:



Blue: two power iterations Green: one power iteration Red: no power iteration

The plot shows the histogram for the errors from the randomized range finder. To be precise, we plot $e_k = \|\mathbf{A} - \mathbf{P}_k\mathbf{A}\|$, where $\mathbf{P}_k$ is the orthogonal projection onto the first $k = 50$ columns of $\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)^q\mathbf{A}\mathbf{G}$, and where $\mathbf{G}$ is a Gaussian random matrix.

Randomized low rank approximation: Theory

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\,\mathbf{R}$. `[Q, R] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$. `U = Q * Uhat`

One can prove that with minimal *oversampling*, close to optimal errors are attained:

Randomized low rank approximation: Theory

Goal: Given an $m \times n$ matrix $\mathbf{A}$, compute an approximate rank-k SVD $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$.

Algorithm:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$. `G = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{G}$. `Y = A * G`
3. Form an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\,\mathbf{R}$. `[Q, R] = qr(Y)`
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$. `B = Q' * A`
5. Compute the SVD of the small matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
6. Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$. `U = Q * Uhat`

One can prove that with minimal *oversampling*, close to optimal errors are attained:

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

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

and

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

Randomized low rank approximation: Computational costs

Case 1 — A is given as an array of numbers that fits in RAM ("small matrix"):

Classical methods (e.g. Golub-Businger) have cost $O(mnk)$. The basic randomized method described also has $O(mnk)$ cost, but with a lower pre-factor (and sometimes lower accuracy). However, the cost can be reduced to $O(mn\log(k))$, by replacing the Gaussian embedding by a structured one; for instance, a sub-sampled randomized Fourier transform (SRFT), which can be applied rapidly using variations of the FFT.

Randomized low rank approximation: Computational costs

Case 1 — A is given as an array of numbers that fits in RAM ("small matrix"):

Classical methods (e.g. Golub-Businger) have cost $O(mnk)$. The basic randomized method described also has $O(mnk)$ cost, but with a lower pre-factor (and sometimes lower accuracy). However, the cost can be reduced to $O(mn\log(k))$, by replacing the Gaussian embedding by a structured one; for instance, a sub-sampled randomized Fourier transform (SRFT), which can be applied rapidly using variations of the FFT.

- The algorithm must be modified a bit beside replacing the random matrix.
- The SRFT leads to large speed-ups *for moderate matrix sizes.*
  For instance, for $m = n = 4000$, and $k \sim 10^2$, we observe about $\times 5$ speedup.
- In practice, accuracy is very similar to what you get from Gaussian random matrices.
- Theory is still quite weak.
- Many different "structured random projections" have been proposed: sub-sampled Hadamard transform, chains of Givens rotations, sparse projections, etc.

References: Ailon & Chazelle (2006); Liberty, Rokhlin, Tygert, and Woolfe (2006); Halko, Martinsson, Tropp (2011); Clarkson & Woodruff (2013).
Much subsequent work — "Fast Johnson-Lindenstrauss transform."

Randomized low rank approximation: Computational costs

Case 1 — A is given as an array of numbers that fits in RAM ("small matrix"):

Classical methods (e.g. Golub-Businger) have cost $O(mnk)$. The basic randomized method described also has $O(mnk)$ cost, but with a lower pre-factor (and sometimes lower accuracy). However, the cost can be reduced to $O(mn\log(k))$, by replacing the Gaussian embedding by a structured one; for instance, a sub-sampled randomized Fourier transform (SRFT), which can be applied rapidly using variations of the FFT.

Case 2 — A is given as an array of numbers on disk ("large matrix"):

In this case, the relevant metric is memory access. Randomized methods access **A** via sweeps over the entire matrix. With slight modifications, the randomized method can be executed in a *single pass* over the matrix. High accuracy can be attained with a small number of passes (say two, three, four).

(In contrast, classical (deterministic) methods require "random" access to matrix elements...)

Randomized low rank approximation: Computational costs

Case 1 — A is given as an array of numbers that fits in RAM ("small matrix"):

Classical methods (e.g. Golub-Businger) have cost $O(mnk)$. The basic randomized method described also has $O(mnk)$ cost, but with a lower pre-factor (and sometimes lower accuracy). However, the cost can be reduced to $O(mn\log(k))$, by replacing the Gaussian embedding by a structured one; for instance, a sub-sampled randomized Fourier transform (SRFT), which can be applied rapidly using variations of the FFT.

Case 2 — A is given as an array of numbers on disk ("large matrix"):

In this case, the relevant metric is memory access. Randomized methods access **A** via sweeps over the entire matrix. With slight modifications, the randomized method can be executed in a *single pass* over the matrix. High accuracy can be attained with a small number of passes (say two, three, four).

(In contrast, classical (deterministic) methods require "random" access to matrix elements...)

Case 3 — A and $\mathbf{A}^*$ can be applied fast ("structured matrix"):

Think of **A** sparse, or sparse in the Fourier domain, or amenable to the Fast Multipole Method, etc. The classical competitor is in this case "Krylov methods". Randomized methods tend to be more robust, and easier to implement in massively parallel environments. They are more easily blocked to reduce communication. However, Krylov methods sometimes lead to higher accuracy.

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

Single pass algorithms

Problem formulation: You seek to build a low rank approximation to a matrix **A** under the following two constraints:

1. You can view each matrix element only once.
2. You cannot specify the order in which the elements are viewed.

The idea is that the matrix is huge, so that it cannot be stored.

This is *very* restrictive!

To the best of my knowledge, no deterministic algorithms can do this.

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix **A**:

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix **A**:

Input: An $n \times n$ symmetric matrix **A**, and a target rank k.

Output: Rank-k factors **U** and **D** that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix **G**.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix **Q** such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.
4. Project **A** down to $\mathbf{C} = \mathbf{Q}^*\mathbf{AQ}$.
5. Compute the EVD of **C** (which is small): $\mathbf{C} = \hat{\mathbf{U}}\mathbf{D}\hat{\mathbf{U}}^*$.
6. Map back to original space $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

We see that **A** is visited twice, in the computations highlighted in red.

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix **A**:

Input: An $n \times n$ symmetric matrix **A**, and a target rank k.

Output: Rank-k factors **U** and **D** that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix **G**.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix **Q** such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix $\mathbf{A}$:

Input: An $n \times n$ symmetric matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$ and $\mathbf{D}$ that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

We claimed earlier that the columns of $\mathbf{Q}$ approximately span col($\mathbf{A}$), so

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

Since $\mathbf{A}$ is symmetric, we also have $\mathbf{A} \approx \mathbf{AQQ}^*$, so

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{AQQ}^* = \mathbf{QCQ}^*,$$

where

$$\mathbf{C} := \mathbf{Q}^*\mathbf{AQ}.$$

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix $\mathbf{A}$:

Input: An $n \times n$ symmetric matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$ and $\mathbf{D}$ that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

We claimed earlier that the columns of $\mathbf{Q}$ approximately span col($\mathbf{A}$), so

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

Since $\mathbf{A}$ is symmetric, we also have $\mathbf{A} \approx \mathbf{AQQ}^*$, so

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{AQQ}^* = \mathbf{QCQ}^*,$$

where

$$\mathbf{C} := \mathbf{Q}^*\mathbf{AQ}.$$

Right multiply the definition of $\mathbf{C}$ by $\mathbf{Q}^*\mathbf{G}$ to get

$$\mathbf{CQ}^*\mathbf{G} = \mathbf{Q}^*\mathbf{AQQ}^*\mathbf{G} \approx$$

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix $\mathbf{A}$:

Input: An $n \times n$ symmetric matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$ and $\mathbf{D}$ that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

We claimed earlier that the columns of $\mathbf{Q}$ approximately span col($\mathbf{A}$), so

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

Since $\mathbf{A}$ is symmetric, we also have $\mathbf{A} \approx \mathbf{AQQ}^*$, so

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{AQQ}^* = \mathbf{QCQ}^*,$$

where

$$\mathbf{C} := \mathbf{Q}^*\mathbf{AQ}.$$

Right multiply the definition of $\mathbf{C}$ by $\mathbf{Q}^*\mathbf{G}$ to get

$$\mathbf{CQ}^*\mathbf{G} = \mathbf{Q}^*\mathbf{AQQ}^*\mathbf{G} \approx \{\text{Use that } \mathbf{AQQ}^* \approx \mathbf{A}\} \approx \mathbf{Q}^*\mathbf{AG} =$$

Single pass algorithms: The symmetric case

Consider a randomized algorithm for computing the EVD of a *symmetric* matrix **A**:

Input: An $n \times n$ symmetric matrix **A**, and a target rank k.

Output: Rank-k factors **U** and **D** that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix **G**.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix **Q** such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

We claimed earlier that the columns of **Q** approximately span col(**A**), so

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

Since **A** is symmetric, we also have $\mathbf{A} \approx \mathbf{AQQ}^*$, so

$$\mathbf{A} \approx \mathbf{QQ}^*\mathbf{AQQ}^* = \mathbf{QCQ}^*,$$

where

$$\mathbf{C} := \mathbf{Q}^*\mathbf{AQ}.$$

Right multiply the definition of **C** by $\mathbf{Q}^*\mathbf{G}$ to get

$$\mathbf{CQ}^*\mathbf{G} = \mathbf{Q}^*\mathbf{AQQ}^*\mathbf{G} \approx \{\text{Use that } \mathbf{AQQ}^* \approx \mathbf{A}\} \approx \mathbf{Q}^*\mathbf{AG} = \mathbf{Q}^*\mathbf{Y}.$$

Observe that the quantities in red are known and can be formed inexpensively.

As a consequence, we can determine **C** by solving the matrix equation:

$$\underset{k \times k}{\mathbf{C}} \quad \underset{k \times k}{(\mathbf{Q}^*\mathbf{G})} = \underset{k \times k}{(\mathbf{Q}^*\mathbf{Y})}.$$

Single pass algorithms: The symmetric case

Input: An $n \times n$ symmetric matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$ and $\mathbf{D}$ that form an approximate EVD $\mathbf{A} \approx \mathbf{UDU}^*$

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.
4. Solve the matrix equation $\mathbf{C}\,(\mathbf{Q}^*\mathbf{Y}) = (\mathbf{Q}^*\mathbf{G})$ for $\mathbf{C}$, enforcing $\mathbf{C} = \mathbf{C}^*$.
5. Compute the EVD of $\mathbf{C}$ (which is small): $\mathbf{C} = \hat{\mathbf{U}}\mathbf{D}\hat{\mathbf{U}}^*$.
6. Map back to original space $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Note: In practice, over-sampling is essential. By creating a significantly over-determined system for $\mathbf{C}$, you significantly improve the likelihood of recovering a good approximation to $\mathbf{A}$. It additionally improves numerical stability.

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. . . . ?
2. . . . ?
3. . . . ?
4. . . . ?
5. . . . ?
6. . . . ?

Our old approach started:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

While the information in $\mathbf{G}$ and $\mathbf{Y}$ contains everything we need for a symmetric matrix, this simply is not true for a general matrix.

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. . . . ?
2. . . . ?
3. . . . ?
4. . . . ?
5. . . . ?
6. . . . ?

Our old approach started:

1. Draw an $n \times k$ Gaussian random matrix $\mathbf{G}$.
2. Compute an $n \times k$ sample matrix $\mathbf{Y} = \mathbf{AG}$.
3. Compute an $n \times k$ ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{QQ}^*\mathbf{Y}$.

While the information in $\mathbf{G}$ and $\mathbf{Y}$ contains everything we need for a symmetric matrix, this simply is not true for a general matrix.

We have no idea about the directions of the right singular vectors!

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{A}\mathbf{G}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$. *This can be done in one pass!!*
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{AG}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$. *This can be done in one pass!!*
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.

The columns of $\mathbf{Q}_{\mathrm{c}}$ and $\mathbf{Q}_{\mathrm{r}}$ form approximate bases for the column and row spaces of $\mathbf{A}$, respectively, so

$$\mathbf{A} \approx \mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* = \mathbf{Q}_{\mathrm{c}}\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*,$$

where

$$\mathbf{C} := \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}. \tag{5}$$

First right multiply (5) by $\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}$ to obtain

$$\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} \approx \tag{6}$$

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{AG}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$. *This can be done in one pass!!*
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.

The columns of $\mathbf{Q}_{\mathrm{c}}$ and $\mathbf{Q}_{\mathrm{r}}$ form approximate bases for the column and row spaces of $\mathbf{A}$, respectively, so

$$\mathbf{A} \approx \mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* = \mathbf{Q}_{\mathrm{c}}\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*,$$

where

$$\mathbf{C} := \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}. \tag{5}$$

First right multiply (5) by $\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}$ to obtain

$$\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} \approx \{\text{Use that } \mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* \approx \mathbf{A}\} \approx \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{G}_{\mathrm{c}} = \tag{6}$$

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{AG}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$. *This can be done in one pass!!*
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.

The columns of $\mathbf{Q}_{\mathrm{c}}$ and $\mathbf{Q}_{\mathrm{r}}$ form approximate bases for the column and row spaces of $\mathbf{A}$, respectively, so

$$\mathbf{A} \approx \mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* = \mathbf{Q}_{\mathrm{c}}\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*,$$

where

$$\mathbf{C} := \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}. \tag{5}$$

First right multiply (5) by $\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}$ to obtain

$$\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} \approx \{\text{Use that } \mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* \approx \mathbf{A}\} \approx \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{Y}_{\mathrm{c}}. \tag{6}$$

Single pass algorithms: The general case

Now consider a *general* $m \times n$ matrix $\mathbf{A}$. (Not necessarily symmetric.)

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{AG}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$. *This can be done in one pass!!*
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.

The columns of $\mathbf{Q}_{\mathrm{c}}$ and $\mathbf{Q}_{\mathrm{r}}$ form approximate bases for the column and row spaces of $\mathbf{A}$, respectively, so

$$\mathbf{A} \approx \mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* = \mathbf{Q}_{\mathrm{c}}\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*,$$

where

$$\mathbf{C} := \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}. \tag{5}$$

First right multiply (5) by $\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}$ to obtain

$$\mathbf{C}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}} \approx \{\text{Use that } \mathbf{A}\mathbf{Q}_{\mathrm{r}}\mathbf{Q}_{\mathrm{r}}^* \approx \mathbf{A}\} \approx \mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{G}_{\mathrm{c}} = \mathbf{Q}_{\mathrm{c}}^*\mathbf{Y}_{\mathrm{c}}. \tag{6}$$

Next left multiply (5) by $\mathbf{G}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{c}}$ to obtain

$$\mathbf{G}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{c}}\mathbf{C} = \mathbf{G}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}} \approx \{\text{Use that } \mathbf{Q}_{\mathrm{c}}\mathbf{Q}_{\mathrm{c}}^*\mathbf{A} \approx \mathbf{A}\} \approx \mathbf{G}_{\mathrm{r}}^*\mathbf{A}\mathbf{Q}_{\mathrm{r}} = \mathbf{Y}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{r}}. \tag{7}$$

Finally, define $\mathbf{C}$ as the least-square solution of the two equations

$$\left(\mathbf{G}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{c}}\right)\mathbf{C} = \left(\mathbf{Y}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{r}}\right) \qquad \text{and} \qquad \mathbf{C}\left(\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}\right) = \left(\mathbf{Q}_{\mathrm{c}}^*\mathbf{Y}_{\mathrm{c}}\right).$$

Single pass algorithms: The general case

Input: An $m \times n$ matrix $\mathbf{A}$, and a target rank k.

Output: Rank-k factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ that form an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$

1. Draw two Gaussian random matrices $\mathbf{G}_{\mathrm{c}}$ of size $n \times k$ and $\mathbf{G}_{\mathrm{r}}$ of size $m \times k$.
2. Form two sampling matrices $\mathbf{Y}_{\mathrm{c}} = \mathbf{AG}_{\mathrm{c}}$ and $\mathbf{Y}_{\mathrm{r}} = \mathbf{A}^*\mathbf{G}_{\mathrm{r}}$.

   Step 2 can be executed in one pass over the matrix.
3. Compute two basis matrices $\mathbf{Q}_{\mathrm{c}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{c}})$ and $\mathbf{Q}_{\mathrm{r}} = \mathtt{orth}(\mathbf{Y}_{\mathrm{r}})$.
4. Determine $\mathbf{C}$ by solving $\left(\mathbf{G}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{c}}\right)\mathbf{C} = \left(\mathbf{Y}_{\mathrm{r}}^*\mathbf{Q}_{\mathrm{r}}\right)$ and $\mathbf{C}\left(\mathbf{Q}_{\mathrm{r}}^*\mathbf{G}_{\mathrm{c}}\right) = \left(\mathbf{Q}_{\mathrm{c}}^*\mathbf{Y}_{\mathrm{c}}\right)$ for $\mathbf{C}$.
5. Compute the SVD of $\mathbf{C}$ (which is small): $\mathbf{C} = \hat{\mathbf{U}}\mathbf{D}\hat{\mathbf{V}}^*$.
6. Map back to original space $\mathbf{U} = \mathbf{Q}_{\mathrm{c}}\hat{\mathbf{U}}$ and $\mathbf{V} = \mathbf{Q}_{\mathrm{r}}\hat{\mathbf{V}}$.

Single pass algorithms: Summary

Idea: Build sketches of the row and column spaces in a single pass over the matrix. Then "back out" the missing information via a least squares solvers.

Notes:

- You have no recourse if the rank is higher than you thought!
- Appears to not be compatible with power iteration.
- Implementation details are important:
  - Over sampling. (Do more than in the regular RSVD.)
  - Manage memory constraints. Structured random maps can be helpful.
  - Additional bells and whistles have been proposed, "core samples", etc.

Do not implement the method the way it's presented in these slides, basically . . .

References: *Woolfe, Liberty, Rokhlin, and Tygert (2008), Clarkson and Woodruff (2009), Halko, Martinsson and Tropp (2011), Tropp, Yurtsever, Udell, Cevher (2017), Martinsson and Tropp (2020).*

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

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

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

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

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 $\{1, 2, \ldots, T\}$.
2. Draw a subset of k indices $J = \{t_1, t_2, \ldots, t_k\} \subseteq \{1, 2, \ldots, T\}$.
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 $k \sim \frac{1}{\epsilon^2}$ to attain precision ϵ.

Matrix approximation by sampling: Low rank approximation.

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

Sampling approach:

1. Draw vectors J and I holding k samples from the column and row indices, resp.
2. Form matrices $\mathbf{C}$ and $\mathbf{R}$ consisting of the corresponding columns and rows

$$\mathbf{C} = \mathbf{A}(:, J), \qquad \text{and} \qquad \mathbf{R} = \mathbf{A}(I, :).$$

3. Use as your approximation

$$\begin{array}{ccccc} \bar{\mathbf{A}} & = & \mathbf{C} & \mathbf{U} & \mathbf{R}, \\ m \times n & & m \times k & k \times k & k \times n \end{array}$$

   where $\mathbf{U}$ is computed from information in $\mathbf{A}(I, J)$. (It should be an approximation to the optimal choice $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$. For instance, $\mathbf{U} = \mathbf{A}(I, J)^{-1}$.)

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

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

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

Matrix approximation by sampling: Connections to randomized embedding.

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

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

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

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

Matrix approximation by sampling: Connections to randomized embedding.

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

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

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

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

Matrix approximation by sampling: Connections to randomized embedding.

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

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

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

As we have seen, this in general does not work very well. But it does work well for the class of matrices for which uniform sampling is optimal. *We can turn* $\mathbf{A}$ *into such a matrix!* Let $\mathbf{\Omega}$ 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 $\mathbf{Y} = \mathbf{A\Omega}$, and then form $\mathbf{A}_k = \mathbf{YY}^\dagger\mathbf{A}$.

Choices of random embeddings:

- *Gaussian (or slice of Haar matrix):* Optimal. Leads to $O(mnk)$ overall cost.

- *Subsampled randomized Fourier transform (SRFT):* Indistinguishable from Gaussian in practice. Leads to $O(mn\log(k))$ overall cost. Adversarial counter examples can be built, so supporting theory is weak.

- *Chains of Givens rotations:* Similar profile to an SRFT.

- *Sparse random projections:* Need at least two nonzero entries per row. Works surprisingly well.

- *Additive random projections:* You can use a map with only ± 1 entries.

Matrix approximation by sampling: Key points

- These techniques provide a path forwards for problems where traditional techniques are 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.

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

Interpolative and CUR decompositions

Any $m \times n$ matrix $\mathbf{A}$ of rank k admits a *CUR factorization* of the form

$$\begin{array}{ccccc} \mathbf{A} & = & \mathbf{C} & \mathbf{U} & \mathbf{R}, \\ m \times n & & m \times k & k \times k & k \times n \end{array} \tag{8}$$

where $\mathbf{C} = \mathbf{A}(:, J_{\mathrm{S}})$ holds a subset of the columns of $\mathbf{A}$, and where $\mathbf{R} = \mathbf{A}(I_{\mathrm{S}}, :)$ holds a subset of the rows of $\mathbf{A}$.

Advantages of the CUR factorization include:

- If $\mathbf{A}$ is sparse, then $\mathbf{C}$ and $\mathbf{R}$ are sparse. (Same for "non negative".)
- The CUR factorization requires less storage than other factorizations.
- Knowledge of the index vectors I_{S} and J_{S} is useful for data interpretation.

Note: In practice, matrices under consideration do not have *exact* rank k, so we work with approximate factorizations $\mathbf{A} \approx \mathbf{CUR}$.

Interpolative and CUR decompositions: The "column ID"

Let us start by building a factorization that uses the columns to span the column space. (We will deal with the rows later.)

To be precise, given $\mathbf{A} \in \mathbb{R}^{m \times n}$ of rank k, we build the *"column interpolative decomposition (column ID)"*

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{C}} \; \underset{k \times n}{\mathbf{Z}}, \tag{9}$$

where the matrix $\mathbf{C} = \mathbf{A}(:, J_{\mathrm{S}})$ is given by a subset of the columns of $\mathbf{A}$.

- The fact that (9) *exists* follows immediately from the definition of rank.
- More subtle fact: There exist J_{S} for which the factorization is *well-conditioned* in the sense that $|\mathbf{Z}(i, j)| \leq 1$ for all i, j.
- Finding the best J_{S} is combinatorially hard.
- Basic old fashioned Gram Schmidt typically results in a very good J_{S}.
- By combining Gram Schmidt with a random embedding, you get a killer algorithm!

Interpolative and CUR decompositions: Randomized column ID

Theorem: Let $\mathbf{A}$ be an $m \times n$ matrix of rank k. Suppose:

(1) We have by some means computed a factorization

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{E}} \; \underset{k \times n}{\mathbf{F}}.$$

(2) We have computed a column ID of $\mathbf{F}$, so that

$$\underset{k \times n}{\mathbf{F}} = \underset{k \times k}{\mathbf{F}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Then, *automatically,* we also obtain a column ID for $\mathbf{A}$:

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{A}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Interpolative and CUR decompositions: Randomized column ID

Theorem: Let $\mathbf{A}$ be an $m \times n$ matrix of rank k. Suppose:

(1) We have by some means computed a factorization

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{E}} \; \underset{k \times n}{\mathbf{F}}.$$

(2) We have computed a column ID of $\mathbf{F}$, so that

$$\underset{k \times n}{\mathbf{F}} = \underset{k \times k}{\mathbf{F}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Then, *automatically,* we also obtain a column ID for $\mathbf{A}$:

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{A}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Question: How do you find a $k \times n$ matrix $\mathbf{F}$ such that $\mathbf{A} = \mathbf{EF}$ for some $\mathbf{E}$?

Interpolative and CUR decompositions: Randomized column ID

Theorem: Let $\mathbf{A}$ be an $m \times n$ matrix of rank k. Suppose:

(1) We have by some means computed a factorization

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{E}} \; \underset{k \times n}{\mathbf{F}}.$$

(2) We have computed a column ID of $\mathbf{F}$, so that

$$\underset{k \times n}{\mathbf{F}} = \underset{k \times k}{\mathbf{F}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Then, *automatically,* we also obtain a column ID for $\mathbf{A}$:

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{A}(:, J_{\mathrm{S}})} \; \underset{k \times n}{\mathbf{Z}}.$$

Question: How do you find a $k \times n$ matrix $\mathbf{F}$ such that $\mathbf{A} = \mathbf{EF}$ for some $\mathbf{E}$?

Randomized embedding! Draw a $k \times m$ embedding $\boldsymbol{\Omega}$ and set

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

We do not need to know the factor $\mathbf{E}$*! It just never enters the computation.*

Note: The probability that a set of k columns is sampled is in a certain sense proportional to its spanning volume. This is precisely the property we are after.

Interpolative and CUR decompositions: Randomized column ID

ALGORITHM: BASIC RANDOMIZED ID

Inputs: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p.

Outputs: An index vector J_{S} and a $k \times n$ interpolation matrix $\mathbf{Z}$ such that

$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{A}(:, J_{\mathrm{S}})} \underset{k \times n}{\mathbf{Z}}.$$

(1) Draw a $(k + p) \times m$ random matrix $\mathbf{\Omega}$ from a Gaussian distribution;

(2) $\mathbf{F} = \mathbf{\Omega A}$;

(3) Do k steps of Gram-Schmidt on the columns of $\mathbf{F}$ to form the ID $\mathbf{F} \approx \mathbf{F}(:, J_{\mathrm{S}})\mathbf{Z}$.

Complexity is $O(mnk)$ for a general dense matrix.

Particularly effective when $\mathbf{A}$ is sparse.

Power iteration can be used to enhance the optimality of the selection.

Interpolative and CUR decompositions: Fast randomized column ID

ALGORITHM: FAST RANDOMIZED ID

Inputs: An $m \times n$ matrix $\mathbf{A}$, a target rank k, and an over-sampling parameter p.

Outputs: An index vector J_{S} and a $k \times n$ interpolation matrix $\mathbf{Z}$ such that

$$\underset{m \times n}{\mathbf{A}} \approx \underset{m \times k}{\mathbf{A}(:, J_{\mathrm{S}})} \underset{k \times n}{\mathbf{Z}}.$$

(1) Draw a fast Johnson-Lindenstrauss transform (e.g. an SRFT) $\mathbf{\Omega}$;

(2) $\mathbf{F} = \mathbf{\Omega A}$;

(3) Do k steps of Gram-Schmidt on the columns of $\mathbf{F}$ to form the ID $\mathbf{F} \approx \mathbf{F}(:, J_{\mathrm{S}})\mathbf{Z}$.

Complexity is $O(mn \log k)$ for a general dense matrix.

Interpolative and CUR decompositions: Returning to CUR

Previously, we did a bit of a bait and switch when we shifted from CUR to column ID ...

However, going from column ID to CUR is easy.

Suppose that we have computed a factorization

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{C}} \; \underset{k \times n}{\mathbf{Z}}, \tag{10}$$

where $\mathbf{C} = \mathbf{A}(:, J_{\mathrm{S}})$.

Interpolative and CUR decompositions: Returning to CUR

Previously, we did a bit of a bait and switch when we shifted from CUR to column ID ...

However, going from column ID to CUR is easy.

Suppose that we have computed a factorization

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{C}} \; \underset{k \times n}{\mathbf{Z}}, \tag{10}$$

where $\mathbf{C} = \mathbf{A}(:, J_{\mathrm{S}})$. We can now obtain the CUR in two steps:

1. To obtain the index vector I_{S} pointing to the rows in the CUR, *just do Gram-Schmidt on the rows of* $\mathbf{C}$*!*
2. Set $\mathbf{R} = \mathbf{A}(I_{\mathrm{S}}, :)$.
3. Compute $\mathbf{U} = \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$.

Some comments:

- There are shortcuts for executing Step 3. (E.g., use $\mathbf{U} = \mathbf{A}(I_{\mathrm{S}}, J_{\mathrm{S}})^{-1}$.)
- The CUR factorization is *inherently unstable*. Avoid step 3 if possible!
- The interpolative decomposition is stable, and provides everything you need.
- CUR factorizations can be computed by sampling rows and columns using "leverage scores". Subject of much interest in the literature...

Interpolative and CUR decompositions: Connections to column pivoted QR

Next, let us discuss connections to the column pivoted QR factorization of a matrix.

This will get slightly more technical.

But no worries, only for a couple of slides!

Interpolative and CUR decompositions: Connections to column pivoted QR

Question: How precisely do you get the column ID from the CPQR?

Recall that the Gram-Schmidt process can be described conveniently via the QR factorization. To be precise, after k steps of the Gram-Schmidt process, we have determined a factorization

$$\underset{m\times n}{\mathbf{A}(:,J)} = \underset{m\times k}{\mathbf{Q}_1}\ \Big[\ \underset{k\times k}{\mathbf{R}_{11}}\quad \underset{k\times(n-k)}{\mathbf{R}_{12}}\ \Big] + \Big[\ \underset{m\times k}{\mathbf{0}}\quad \underset{m\times(n-k)}{\mathbf{B}}\ \Big], \tag{11}$$

where J is a permutation of the indices $[1, 2, \ldots, n]$, where $\mathbf{R}_{11}$ is upper triangular, and where $\mathbf{B}$ is a remainder matrix that is small in magnitude. Let us partition the index vector

$$J = [J_{\mathrm{S}} \quad J_{\mathrm{rem}}],$$

so that J_{S} is a vector identifying the k chosen "pivot columns." Then by construction

$$\mathbf{A}(:, J_{\mathrm{S}}) = \mathbf{Q}_1\mathbf{R}_{11}.$$

Now let $\mathbf{P}$ be the permutation matrix for which $\mathbf{A}(:,J) = \mathbf{AP}$. Then (11) can be written

$$\mathbf{AP} = \mathbf{Q}_1\mathbf{R}_{11}\big[\mathbf{I} \quad \mathbf{R}_{11}^{-1}\mathbf{R}_{12}\big] + \big[\mathbf{0} \quad \mathbf{B}\big].$$

Right multiplying by $\mathbf{P}^*$ (recall that $\mathbf{PP}^* = \mathbf{I}$ since $\mathbf{P}$ is unitary) we get

$$\mathbf{A} = \underbrace{\mathbf{A}(:, J_{\mathrm{S}})}_{=:\mathbf{C}}\underbrace{\big[\mathbf{I} \quad \mathbf{R}_{11}^{-1}\mathbf{R}_{12}\big]\mathbf{P}^*}_{=:\mathbf{Z}} + \big[\mathbf{0} \quad \mathbf{B}\big]\mathbf{P}^*.$$

Interpolative and CUR decompositions: Connections to column pivoted QR

Next we will describe how randomization can be used to effectively compute an *full* factorization of a matrix.

Observe that all previous factorizations have been low rank approximations in some sense.

Note: All methods discussed have complexity $O(n^3)$. We are discussing the scaling factor.

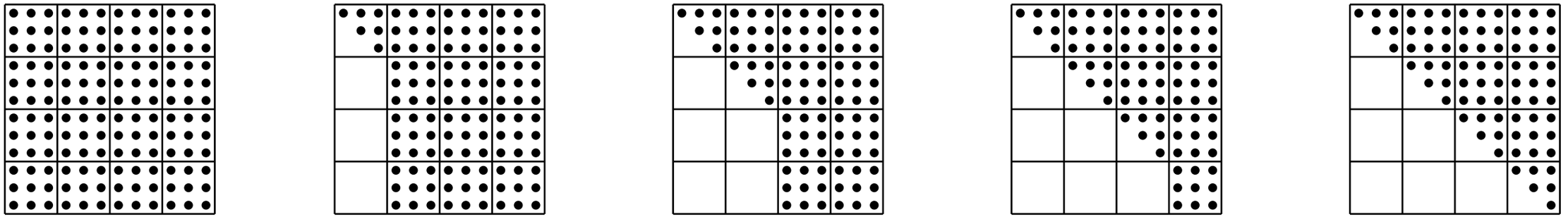
Computing full (or nearly full) rank revealing factorizations of matrices

Given a dense $n \times n$ matrix $\mathbf{A}$, compute a column pivoted QR factorization

$$\underset{n \times n}{\mathbf{A}} \quad \underset{n \times n}{\mathbf{P}} \approx \underset{n \times n}{\mathbf{Q}} \quad \underset{n \times n}{\mathbf{R}},$$

where, as usual, $\mathbf{Q}$ should be ON, $\mathbf{P}$ is a permutation, and $\mathbf{R}$ is upper triangular.

The technique proposed is based on a *blocked* version of classical Householder QR:



$$\mathbf{A}_0 = \mathbf{A} \qquad \mathbf{A}_1 = \mathbf{Q}_1^* \mathbf{A}_0 \mathbf{P}_1 \qquad \mathbf{A}_2 = \mathbf{Q}_2^* \mathbf{A}_1 \mathbf{P}_2 \qquad \mathbf{A}_3 = \mathbf{Q}_3^* \mathbf{A}_2 \mathbf{P}_3 \qquad \mathbf{A}_4 = \mathbf{Q}_4^* \mathbf{A}_3 \mathbf{P}_4$$

Each $\mathbf{P}_j$ is a permutation matrix computed via randomized sampling.

Each $\mathbf{Q}_j$ is a product of Householder reflectors.

The key challenge has been to find good permutation matrices.

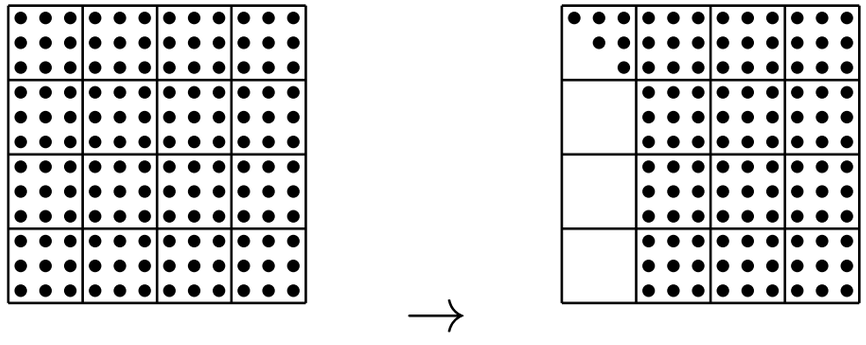
We seek $\mathbf{P}_j$ so that the set of b chosen columns *has maximal spanning volume.*

Perfect for randomized sampling! The likelihood that any block of columns is "hit" by the random vectors is directly proportional to its volume. Perfect optimality is *not* required.

Computing full (or nearly full) rank revealing factorizations of matrices

How to do block pivoting using randomization:

Let **A** be of size $m \times n$, and let b be a block size.



A $\qquad\qquad$ **Q*****AP**

Q is a product of b Householder reflectors.

P is a permutation matrix that moves b "pivot" columns to the leftmost slots.

We seek **P** so that the set of chosen columns *has maximal spanning volume.*

Draw a Gaussian random matrix $\mathbf{\Omega}$ of size $b \times m$ and form

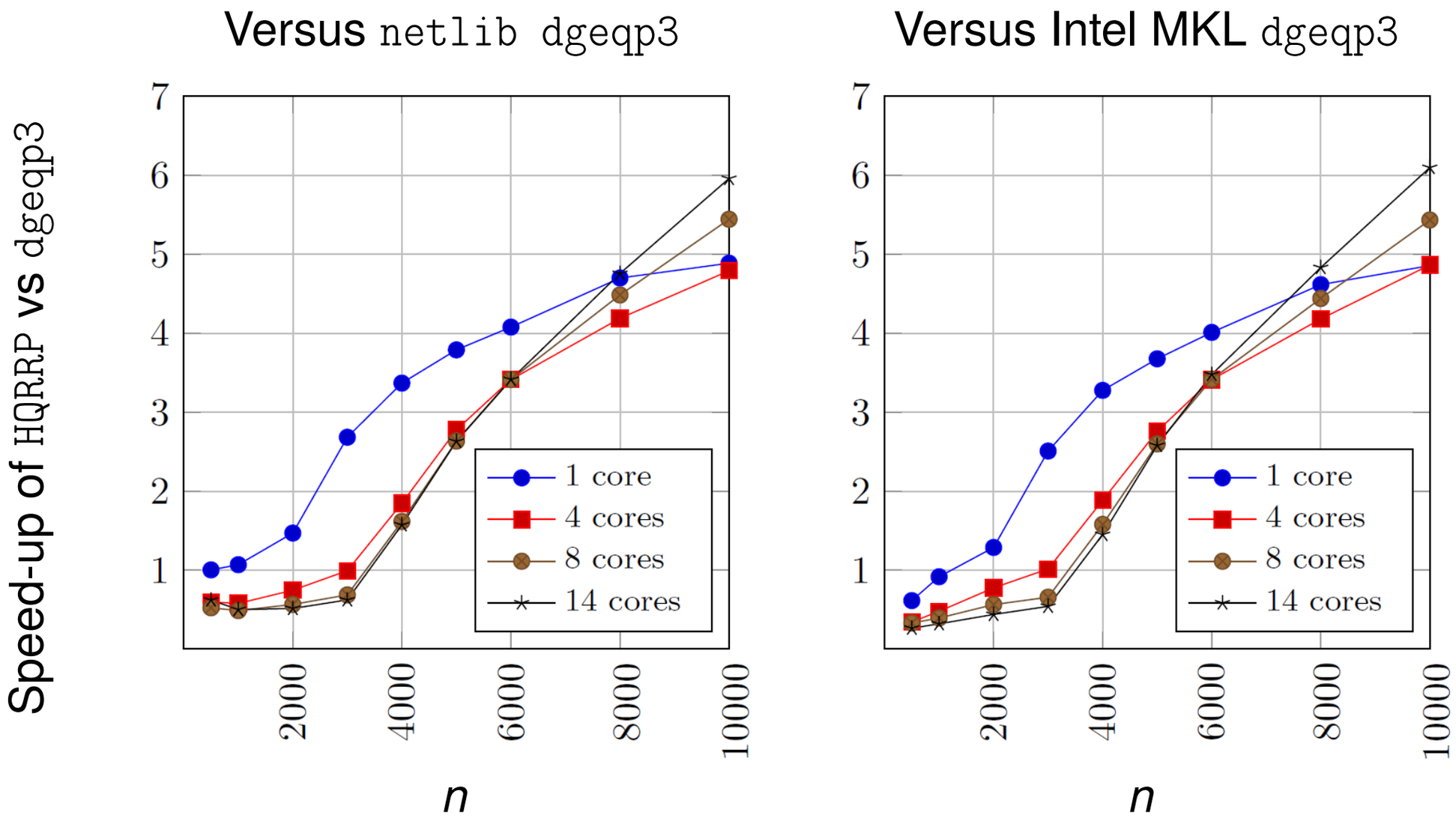
$$\underset{b \times n}{\mathbf{Y}} = \underset{b \times m}{\mathbf{\Omega}} \; \underset{m \times n}{\mathbf{A}}$$

The rows of **Y** are random linear combinations of the rows of **A**.

Then compute the pivot matrix **P** for the first block by executing traditional column pivoting on the small matrix **Y**:

$$\underset{b \times n}{\mathbf{Y}} \; \underset{n \times n}{\mathbf{P}} = \underset{b \times b}{\mathbf{Q}_{\text{trash}}} \; \underset{b \times n}{\mathbf{R}_{\text{trash}}}$$

Computing full (or nearly full) rank revealing factorizations of matrices



Speedup attained by our randomized algorithm `HQRRP` 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. Available at: `https://github.com/flame/hqrrp/`

Computing full (or nearly full) rank revealing factorizations of matrices

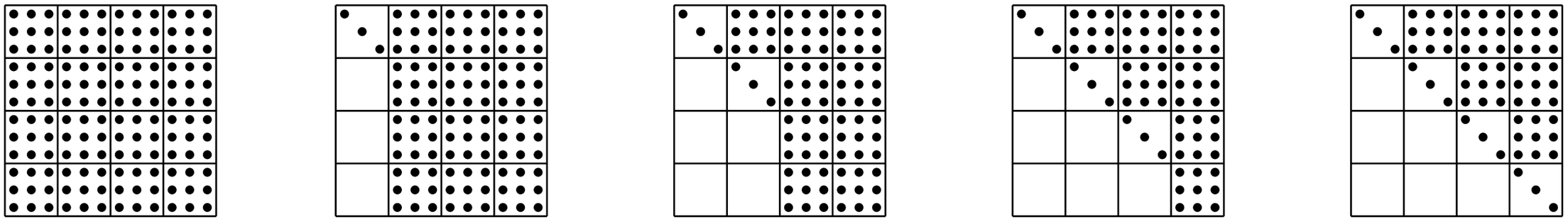
Given a dense $n \times n$ matrix $\mathbf{A}$, compute a factorization

$$\underset{n \times n}{\mathbf{A}} = \underset{n \times n}{\mathbf{U}} \quad \underset{n \times n}{\mathbf{T}} \quad \underset{n \times n}{\mathbf{V}^*},$$

where $\mathbf{T}$ is upper triangular, $\mathbf{U}$ and $\mathbf{V}$ are unitary.

Observe: More general than CPQR since we used to insist that $\mathbf{V}$ be a permutation.

The technique proposed is based on a blocked version of classical Householder QR:



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

Both $\mathbf{U}_j$ and $\mathbf{V}_j$ are (mostly...) products of b Householder reflectors.

Our objective is in each step to find an approximation *to the linear subspace* spanned by the b dominant singular vectors of a matrix. The randomized range finder is perfect for this, especially when a small number of power iterations are performed. Easier and more natural than choosing pivoting vectors.

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Overdetermined linear regression

Suppose that $\mathbf{A} \in \mathbb{R}^{m \times n}$, where $m \gg n$, and that we seek to solve the linear system

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

in a least squares sense. Complexity of standard solvers: $O(mn^2)$

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Overdetermined linear regression

Suppose that $\mathbf{A} \in \mathbb{R}^{m \times n}$, where $m \gg n$, and that we seek to solve the linear system

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

in a least squares sense. Complexity of standard solvers: $O(mn^2)$

Method proposed by Rokhlin and Tygert (PNAS 2008): Form a "sketched equation"

$$\mathbf{\Omega}^* \mathbf{Ax} = \mathbf{\Omega}^* \mathbf{b}$$

where $\mathbf{\Omega}$ is an $m \times \ell$ SRFT. Compute QR factorisation of the new coefficient matrix

$$\mathbf{\Omega}^* \mathbf{A} = \mathbf{QRP}^*.$$

Form a preconditioner

$$\mathbf{M} = \mathbf{RP}^*.$$

Solve the preconditioned linear system

$$\left(\mathbf{AM}^{-1}\right) \underbrace{\left(\mathbf{Mx}\right)}_{=:\mathbf{y}} = \mathbf{b}$$

for the new unknown $\mathbf{y}$. Complexity of randomized solver: $O\big((\log(n) + \log(1/\varepsilon))mn + n^3\big)$.

Later improvements include BLENDENPIK by Avron, Maymounkov, Toledo (2010).

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Overdetermined linear regression

Suppose that $\mathbf{A} \in \mathbb{R}^{m \times n}$, where $m \gg n$, and that we seek to solve the linear system

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

in a least squares sense. Complexity of standard solvers: $O(mn^2)$

Important: What we saw was an example of a *randomized preconditioner*.

You continue the iteration until the residual $\|\mathbf{Ax}_k - \mathbf{b}\|$ goes below a given tolerance.

The accuracy of the solution is *for sure* below the specified tolerance.

The *runtime* is a random variable.

Very close to a *Las Vegas* algorithm.

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Randomized Kaczmarz

The classical Kaczmarz algorithm:

With $\mathbf{A} \in \mathbb{R}^{m \times n}$, we seek to solve $\mathbf{Ax} = \mathbf{b}$ through an iterative procedure.

Given an approximate solution $\mathbf{x}_{\text{old}}$, compute an improved solution $\mathbf{x}_{\text{new}}$ as follows:

(1) Pick a row index $i \in \{1, 2, \ldots, m\}$.

(2) Require that $\mathbf{x}_{\text{new}}$ is picked so that row i of the system is satisfied exactly.

(3) Within the hyperplane defined by (2), pick $\mathbf{x}_{\text{new}}$ as the point closest to $\mathbf{x}_{\text{old}}$.

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Randomized Kaczmarz

The classical Kaczmarz algorithm:

With $\mathbf{A} \in \mathbb{R}^{m \times n}$, we seek to solve $\mathbf{Ax} = \mathbf{b}$ through an iterative procedure.

Given an approximate solution $\mathbf{x}_{\text{old}}$, compute an improved solution $\mathbf{x}_{\text{new}}$ as follows:

(1) Pick a row index $i \in \{1, 2, \ldots, m\}$.

(2) Require that $\mathbf{x}_{\text{new}}$ is picked so that row i of the system is satisfied exactly.

(3) Within the hyperplane defined by (2), pick $\mathbf{x}_{\text{new}}$ as the point closest to $\mathbf{x}_{\text{old}}$.

The resulting formula is $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \dfrac{\mathbf{b}(i) - \left(\mathbf{A}(i,:) \cdot \mathbf{x}_{\text{old}}\right)}{\|\mathbf{A}(i,:)\|^2} \mathbf{A}(i,:)^*$.

Question: How do you pick the row index i in step (1)?

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Randomized Kaczmarz

The classical Kaczmarz algorithm:

With $\mathbf{A} \in \mathbb{R}^{m \times n}$, we seek to solve $\mathbf{Ax} = \mathbf{b}$ through an iterative procedure.

Given an approximate solution $\mathbf{x}_{\text{old}}$, compute an improved solution $\mathbf{x}_{\text{new}}$ as follows:

(1) Pick a row index $i \in \{1, 2, \ldots, m\}$.

(2) Require that $\mathbf{x}_{\text{new}}$ is picked so that row i of the system is satisfied exactly.

(3) Within the hyperplane defined by (2), pick $\mathbf{x}_{\text{new}}$ as the point closest to $\mathbf{x}_{\text{old}}$.

The resulting formula is $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \dfrac{\mathbf{b}(i) - \big(\mathbf{A}(i,:) \cdot \mathbf{x}_{\text{old}}\big)}{\|\mathbf{A}(i,:)\|^2} \mathbf{A}(i,:)^*$.

Question: How do you pick the row index i in step (1)?

Strohmer & Vershynin (2009): Draw i with probability proportional to $\|\mathbf{A}(i,:)\|$.

Theorem: Let $\mathbf{x}_\star$ denote the exact solution to $\mathbf{Ax} = \mathbf{b}$, and let $\mathbf{x}_k$ denote the k'th iterate of the S&V randomized Kaczmarz method. Then

$$\mathbb{E}\big[\|\mathbf{x}_k - \mathbf{x}_\star\|\big] \leq \left(1 - \frac{1}{\kappa(\mathbf{A})^2}\right)^k \|\mathbf{x}_0 - \mathbf{x}_\star\|,$$

where $\kappa(\mathbf{A})$ is the "scaled" condition number $\kappa(\mathbf{A}) = \|\mathbf{A}\|_{\text{F}} \, \|\mathbf{A}^{-1}\|_2$.

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Randomized Kaczmarz

The classical Kaczmarz algorithm:

With $\mathbf{A} \in \mathbb{R}^{m \times n}$, we seek to solve $\mathbf{Ax} = \mathbf{b}$ through an iterative procedure.

Given an approximate solution $\mathbf{x}_{\text{old}}$, compute an improved solution $\mathbf{x}_{\text{new}}$ as follows:

(1) Pick a row index $i \in \{1, 2, \ldots, m\}$.

(2) Require that $\mathbf{x}_{\text{new}}$ is picked so that row i of the system is satisfied exactly.

(3) Within the hyperplane defined by (2), pick $\mathbf{x}_{\text{new}}$ as the point closest to $\mathbf{x}_{\text{old}}$.

The resulting formula is $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \dfrac{\mathbf{b}(i) - \big(\mathbf{A}(i,:) \cdot \mathbf{x}_{\text{old}}\big)}{\|\mathbf{A}(i,:)\|^2} \mathbf{A}(i,:)^*$.

Question: How do you pick the row index i in step (1)?

Gower & Richtarik (2015): Draw an $m \times \ell$ random map $\mathbf{\Omega}$

$$\mathbf{x}_{\text{new}} = \text{argmin}\{\|\mathbf{y} - \mathbf{x}_{\text{old}}\| : \ \mathbf{y} \text{ satisfies } \mathbf{\Omega}^*\mathbf{Ay} = \mathbf{\Omega}^*\mathbf{b}\}.$$

Leads to stronger analysis, and a much richer set of dimension reducing maps.

In particular, it improves practical performance since it enables *blocking*.

Note: An ideal weight for a group of rows would be their spanning volume ...

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Randomized Newton-Schulz

Classical Newton-Schulz for computing $\mathbf{A}^{-1}$: With $\mathbf{A} \in \mathbb{R}^{n\times n}$, we build $\mathbf{B} = \mathbf{A}^{-1}$ through an iterative scheme. Given an approximation $\mathbf{B}_{\text{old}}$, the improved one is

$$\mathbf{B}_{\text{new}} = \mathbf{B}_{\text{old}} - \mathbf{A}\mathbf{B}_{\text{old}}\mathbf{A}.$$

Converges rapidly from a good initial guess. But basin of convergence is not large.

Gower & Richtarik (2019): Find $\mathbf{B} = \mathbf{A}^{-1}$ by solving the equation

$$\mathbf{A}^* = \mathbf{A}^*\mathbf{A}\mathbf{B}. \tag{12}$$

Equation (12) is solved through sketching + iteration: Draw an $m \times \ell$ random map $\mathbf{\Omega}$

$$\mathbf{B}_{\text{new}} = \operatorname{argmin}\{\|\mathbf{M} - \mathbf{B}_{\text{old}}\| : \ \mathbf{M} \text{ satisfies } \mathbf{\Omega}^*\mathbf{A}^* = \mathbf{\Omega}^*\mathbf{A}^*\mathbf{A}\mathbf{M}\}.$$

Equivalent to iteration

$$\mathbf{B}_{\text{new}} = \mathbf{B}_{\text{old}} - \mathbf{A}^*\mathbf{A}\mathbf{\Omega}\left(\mathbf{\Omega}^*\mathbf{A}^*\mathbf{A}\mathbf{A}^*\mathbf{A}\mathbf{\Omega}\right)^{\dagger}\mathbf{\Omega}^*\mathbf{A}^*\left(\mathbf{A}\mathbf{B}_{\text{old}} - \mathbf{I}\right).$$

Detailed error analysis exists. For instance:

The expectation of the error converges exponentially fast, regardless of starting point.

Randomized methods for solving Ax = b: Graph Laplacians

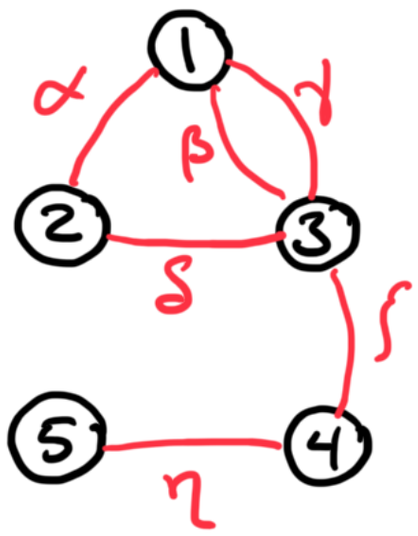
Let us consider a linear system

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

involving a coefficient matrix that is a *graph Laplacian* with n nodes and m edges.

- $\mathbf{A} = \mathbf{A}^* \in \mathbb{R}^{n \times n}$.
- $\mathbf{A}(i,j) \leq 0$ when $i \neq j$.
- $\mathbf{A}(i,i) = -\sum_{j \neq i} \mathbf{A}(i,j)$

We assume that the underlying graph is *connected*, in which case $\mathbf{A}$ has a 1-dimensional nullspace. We enforce that $\sum_i \mathbf{x}(i) = 0$ and $\sum_i \mathbf{b}(i) = 0$ in everything that follows.



(a) A graph with $n = 5$ vertices, and $m = 6$ edges. The conductivities of each edge is marked with a Greek letter.

$$\begin{bmatrix} \alpha+\beta+\gamma & -\alpha & -\beta-\gamma & 0 & 0 \\ -\alpha & \alpha+\delta+\zeta & -\delta & 0 & 0 \\ -\beta-\gamma & -\delta & \beta+\gamma+\delta & -\zeta & 0 \\ 0 & 0 & -\zeta & \zeta+\eta & -\eta \\ 0 & 0 & 0 & -\eta & \eta \end{bmatrix}$$

(b) The 5×5 graph Laplacian matrix associated with the graph shown in (a).

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Graph Laplacians

Let us consider a linear system

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

involving a coefficient matrix that is a *graph Laplacian* with n nodes and m edges.

Standard solution techniques:

- *Multigrid:* Works great for certain classes of matrices.

- *Cholesky:* Compute a decomposition

  $$\mathbf{A} = \mathbf{C}\mathbf{C}^*,$$

  with **C** lower triangular. Always works. Numerically stable (when pivoting is used). Can be expensive since the factor **C** typically has far more non-zero entries than **A**.

- *Incomplete Cholesky:* Compute an approximate factorisation

  $$\mathbf{A} \approx \mathbf{C}\mathbf{C}^*,$$

  where **C** is constrained to be as sparse as **A** (typically the same pattern). Then use CG to solve a system with the preconditioned coefficient matrix $\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-*}$. Can work very well, hard to analyze.

Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$: Graph Laplacians

Let us consider a linear system

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

involving a coefficient matrix that is a *graph Laplacian* with n nodes and m edges.

Randomized solution techniques:

- *Spielman-Teng (2004):* Complexity $O(m \operatorname{poly}(\log n) \log(1/\varepsilon))$.
  Relies on graph theoretical constructs (low-stretch trees, graph sparsification, explicit expander graphs, ...). Important theoretical results.

- *Kyng-Lee-Sachdeva-Spielman (2016):* $O(m (\log n)^2)$.
  Relies on local sampling only. Much closer to a realistic algorithm.

The idea is to build an approximate sparse Cholesky factor that is accurate with high probability. For instance, the 2016 paper proposes to build factors for which

$$\frac{1}{2}\mathbf{A} \preccurlyeq \mathbf{C}\mathbf{C}^* \preccurlyeq \frac{3}{2}\mathbf{A}.$$

When this bound holds, CG converges as $O(\gamma^n)$ with $\gamma = \frac{\sqrt{3}-1}{\sqrt{3}+1} \approx 0.27$.

Sparsity is maintained by performing inexact rank-1 updates in the Cholesky procedure. As a group of edges in the graph is removed, a set of randomly drawn new edges are added, in a way that is correct *in expectation.*

Randomized methods for solving Ax = b: Kernel ridge regression

Task: We are given a set of pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ are data points, and where y_i are corresponding labels. We seek to build a function $f : \mathbb{R}^d \to \mathbb{R}$ such that

$$y_i \approx f(\boldsymbol{x}_i)$$

for every point in the training set. The objective is to predict the label for any new unseen data point $\boldsymbol{x}$.

Methodology: Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a kernel function that measures how similar a pair of points are, scaled so that

$$k(\boldsymbol{x}, \boldsymbol{y}) \approx 1 \qquad \text{means } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ are similar,}$$
$$k(\boldsymbol{x}, \boldsymbol{y}) \approx 0 \qquad \text{means } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ are uncorrelated.}$$

It is the job of the modeler to provide a "good" kernel function.

We then approximate f using the formula $f(\boldsymbol{x}) = \sum_{i=1}^n k(\boldsymbol{x}, \boldsymbol{x}_i)\, \alpha_i$, where the *weights* $\{\alpha_i\}_{i=1}^n$ are computed using the formula $\boldsymbol{\alpha} = \left(\mathbf{K} + \lambda n \mathbf{I}\right)^{-1} \mathbf{y}$, where $\mathbf{K}$ is the $n \times n$ matrix with entries $k(\boldsymbol{x}_i, \boldsymbol{x}_i)$. The number λ is a regularization parameter.

Challenge: $\mathbf{K}$ is very large, and computing an individual entry can be expensive.

Randomized solution: Draw an in index vector $\mathbf{J} \subset \{1, 2, \ldots, n\}$ holding k sampled indices, and replace $\mathbf{K}$ by the formula

$$\mathbf{K}_{\text{approx}} = \mathbf{K}(:, J)\, \mathbf{K}(J, J)^{\dagger}\, \mathbf{K}(J, :).$$

Randomized iterative solvers is a *very* active area: Recent and current work by H. Avron, P. Drineas, L.-H. Lim, M. Mahoney, D. Needell, V. Rokhlin, S. Toledo, J. Tropp, R. Ward, J. Weare, and many more.

Outline:

1. Randomized embeddings.
2. Low rank approximation — review of classical methods.
3. Randomized low rank approximation.
4. Single pass algorithms.
5. Matrix approximation by sampling.
6. CUR and interpolative decompositions.
7. Randomized methods for solving $\mathbf{Ax} = \mathbf{b}$.
8. Analysis of randomized low rank approximation.

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

Question: What is the error $e_k = \|\mathbf{A} - \mathbf{UDV}^*\|$? (Recall that $e_k = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$.)

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

Question: What is the error $e_k = \|\mathbf{A} - \mathbf{UDV}^*\|$? (Recall that $e_k = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$.)

Eckart-Young theorem: e_k is bounded from below by the singular value σ_{k+1} of $\mathbf{A}$.

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

Question: What is the error $e_k = \|\mathbf{A} - \mathbf{UDV}^*\|$? (Recall that $e_k = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$.)

Eckart-Young theorem: e_k is bounded from below by the singular value σ_{k+1} of $\mathbf{A}$.

Question: Is e_k close to σ_{k+1}?

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

Question: What is the error $e_k = \|\mathbf{A} - \mathbf{UDV}^*\|$? (Recall that $e_k = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$.)

Eckart-Young theorem: e_k is bounded from below by the singular value σ_{k+1} of $\mathbf{A}$.

Question: Is e_k close to σ_{k+1}?

Answer: Lamentably, no. The expectation of $\frac{e_k}{\sigma_{k+1}}$ is large, and has very large variance.

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

Question: What is the error $e_k = \|\mathbf{A} - \mathbf{UDV}^*\|$? (Recall that $e_k = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$.)

Eckart-Young theorem: e_k is bounded from below by the singular value σ_{k+1} of **A**.

Question: Is e_k close to σ_{k+1}?

Answer: Lamentably, no. The expectation of $\frac{e_k}{\sigma_{k+1}}$ is large, and has very large variance.

Remedy: Over-sample *slightly*. Compute $k+p$ samples from the range of **A**.

It turns out that $p = 5$ or 10 is often sufficient. $p = k$ is almost always more than enough.

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

Bound on the expectation of the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AG})$.

If $p \geq 2$, then

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

and

$$\mathbb{E}\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq$$

Bound on the expectation of the error for Gaussian test matrices

Let **A** denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let **G** denote an $n \times (k+p)$ Gaussian matrix.

Let **Q** denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AG})$.

If $p \geq 2$, then

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

and

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

Ref: Halko, Martinsson, Tropp, 2009 & 2011

Observations:

- The error depends only on the singular values of **A**.
- The error does not depend on the leading k singular values $\{\sigma_j\}_{j=1}^{k}$ at all.

Large deviation bound for the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{AG})$.

If $p \geq 4$, and u and t are such that $u \geq 1$ and $t \geq 1$, then

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + t\sqrt{\frac{3k}{p+1}} + u\,t\,\frac{e\sqrt{k+p}}{p+1}\right)\sigma_{k+1} + \frac{t\,e\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

except with probability at most $2\,t^{-p} + e^{-u^2/2}$.

Ref: Halko, Martinsson, Tropp, 2009 & 2011; Martinsson, Rokhlin, Tygert (2006)

---

u and t parameterize "bad" events — large u, t is bad, but unlikely.

Certain choices of t and u lead to simpler results. For instance,

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + 16\sqrt{1 + \frac{k}{p+1}}\right)\sigma_{k+1} + 8\frac{\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability at most $3\,e^{-p}$.

Large deviation bound for the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{AG})$.

If $p \geq 4$, and u and t are such that $u \geq 1$ and $t \geq 1$, then

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + t\sqrt{\frac{3k}{p+1}} + u\,t\,\frac{e\sqrt{k+p}}{p+1}\right)\,\sigma_{k+1} + \frac{t\,e\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

except with probability at most $2\,t^{-p} + e^{-u^2/2}$.

Ref: Halko, Martinsson, Tropp, 2009 & 2011; Martinsson, Rokhlin, Tygert (2006)

---

u and t parameterize "bad" events — large u, t is bad, but unlikely.

Certain choices of t and u lead to simpler results. For instance,

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \left(1 + 6\sqrt{(k+p)\cdot p\log p}\right)\,\sigma_{k+1} + 3\sqrt{k+p}\left(\sum_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability at most $3\,p^{-p}$.

Let us look at the error bound a little closer:

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

Case 1 — the singular values decay rapidly: If (σ_j) decays sufficiently rapidly that $\left(\sum_{j>k}\sigma_j^2\right)^{1/2} \approx \sigma_{k+1}$, then we are fine — a minimal amount of over-sampling (say $p = 5$ or $p = k$) drives the error down close to the theoretically minimal value.

Let us look at the error bound a little closer:

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

Case 1 — the singular values decay rapidly: If (σ_j) decays sufficiently rapidly that $\left(\sum_{j>k} \sigma_j^2\right)^{1/2} \approx \sigma_{k+1}$, then we are fine — a minimal amount of over-sampling (say $p = 5$ or $p = k$) drives the error down close to the theoretically minimal value.

Case 2 — the singular values do not decay rapidly: In the worst case, we have

$$\left(\sum_{j>k} \sigma_j^2\right)^{1/2} \sim \sqrt{n-k}\, \sigma_{k+1}.$$

If n is large, and σ_{k+1}/σ_1 is not that small, we could lose all accuracy.

Let us look at the error bound a little closer:

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

Case 1 — the singular values decay rapidly: If (σ_j) decays sufficiently rapidly that $\left(\sum_{j>k}\sigma_j^2\right)^{1/2} \approx \sigma_{k+1}$, then we are fine — a minimal amount of over-sampling (say $p = 5$ or $p = k$) drives the error down close to the theoretically minimal value.

Case 2 — the singular values do not decay rapidly: In the worst case, we have

$$\left(\sum_{j>k}\sigma_j^2\right)^{1/2} \sim \sqrt{n-k}\,\sigma_{k+1}.$$

If n is large, and σ_{k+1}/σ_1 is not that small, we could lose all accuracy.

This is a common situation when you analyze noisy data. If there is, say, 5% noise so that $\sigma_j \approx 0.05\|\mathbf{A}\|$ for $j > k$, then $\left(\sum_{j=k+1}^{\min(m,n)}\sigma_j^2\right)^{1/2} \approx 0.05\|\mathbf{A}\|\sqrt{n}$*!*

Power method for improving accuracy:

The error depends on how quickly the singular values decay.

The faster the singular values decay — the stronger the relative weight of the dominant modes in the samples.

Idea: The matrix $(\mathbf{A}\,\mathbf{A}^*)^q\,\mathbf{A}$ has the same left singular vectors as $\mathbf{A}$, and its singular values are

$$\sigma_j((\mathbf{A}\,\mathbf{A}^*)^q\,\mathbf{A}) = (\sigma_j(\mathbf{A}))^{2\,q+1}.$$

Much faster decay — so let us use the sample matrix

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

instead of

$$\mathbf{Y} = \mathbf{A}\,\mathbf{G}.$$

References: Paper by Rokhlin, Szlam, Tygert (2008). Suggestions by Ming Gu. Also similar to "block power method," and "block Lanczos."

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

- Detailed (and, we believe, close to sharp) error bounds have been proven.
  For instance, with $\mathbf{A}^{\text{computed}} = \mathbf{UDV}^*$, the expectation of the error satisfies:

$$\mathbb{E}\left[\|\mathbf{A} - \mathbf{A}^{\text{computed}}\|\right] \leq \left(1 + 4\sqrt{\frac{2\,\min(m,n)}{k-1}}\right)^{1/(2q+1)} \sigma_{k+1}(\mathbf{A}). \tag{13}$$

  Reference: Halko, Martinsson, Tropp (2011).

- The improved accuracy from the modified scheme comes at a cost;
  $2q + 1$ passes over the matrix are required instead of 1.
  However, q can often be chosen quite small in practice, $q = 2$ or $q = 3$, say.

- The bound (13) assumes exact arithmetic.
  To handle round-off errors, variations of subspace iterations can be used.
  These are entirely numerically stable and achieve the same error bound.

Bound on the expectation of the error for Gaussian test matrices

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Let k denote a target rank and let p denote an over-sampling parameter.

Let $\mathbf{G}$ denote an $n \times (k+p)$ Gaussian matrix.

Let $\mathbf{Q}$ denote the $m \times (k+p)$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AG})$.

If $p \geq 2$, then

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

and

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

Ref: Halko, Martinsson, Tropp, 2009 & 2011

Proofs — Overview:

Let **A** denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
Let k denote a target rank and let p denote an over-sampling parameter. Set $\ell = k + p$.
Let $\boldsymbol{\Omega}$ denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{A}\boldsymbol{\Omega})$.

We seek to bound the error $e_k = e_k(\mathbf{A}, \boldsymbol{\Omega}) = \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$, which is a random variable.

1. Make no assumption on $\boldsymbol{\Omega}$. Construct a deterministic bound of the form

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \cdots \mathbf{A} \cdots \boldsymbol{\Omega} \cdots$$

Proofs — Overview:

Let $\mathbf{A}$ denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
Let k denote a target rank and let p denote an over-sampling parameter. Set $\ell = k + p$.
Let $\mathbf{\Omega}$ denote an $n \times \ell$ "test matrix", and let $\mathbf{Q}$ denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{A\Omega})$.

We seek to bound the error $e_k = e_k(\mathbf{A}, \mathbf{\Omega}) = \|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|$, which is a random variable.

1. Make no assumption on $\mathbf{\Omega}$. Construct a deterministic bound of the form

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\| \leq \cdots \mathbf{A} \cdots \mathbf{\Omega} \cdots$$

2. Assume that $\mathbf{\Omega}$ is drawn from a normal Gaussian distribution.

   Take expectations of the deterministic bound to attain a bound of the form

$$\mathbb{E}\left[\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|\right] \leq \cdots \mathbf{A} \cdots$$

Proofs — Overview:

Let **A** denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
Let k denote a target rank and let p denote an over-sampling parameter. Set $\ell = k + p$.
Let $\boldsymbol{\Omega}$ denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{A}\boldsymbol{\Omega})$.

We seek to bound the error $e_k = e_k(\mathbf{A}, \boldsymbol{\Omega}) = \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|$, which is a random variable.

1. Make no assumption on $\boldsymbol{\Omega}$. Construct a deterministic bound of the form

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \cdots \mathbf{A} \cdots \boldsymbol{\Omega} \cdots$$

2. Assume that $\boldsymbol{\Omega}$ is drawn from a normal Gaussian distribution.
   Take expectations of the deterministic bound to attain a bound of the form

$$\mathbb{E}\big[\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|\big] \leq \cdots \mathbf{A} \cdots$$

3. Assume that $\boldsymbol{\Omega}$ is drawn from a normal Gaussian distribution.
   Take expectations of the deterministic bound conditioned on "bad behavior" in $\boldsymbol{\Omega}$ to get that

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \cdots \mathbf{A} \cdots$$

   holds with probability at least $\cdots$.

Part 1 (out of 3) — deterministic bound:

Let **A** denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
Let k denote a target rank and let p denote an over-sampling parameter. Set $\ell = k + p$.
Let $\boldsymbol{\Omega}$ denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{A}\boldsymbol{\Omega})$.

Partition the SVD of **A** as follows:

$$\mathbf{A} = \mathbf{U} \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \mathbf{D}_1 & \\ & \mathbf{D}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} \\ \end{matrix} \\ \begin{bmatrix} \mathbf{V}_1^* \\ \mathbf{V}_2^* \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} k \\ \end{matrix} \\ \begin{matrix} n-k \\ \end{matrix} \end{array}$$

Define $\boldsymbol{\Omega}_1$ and $\boldsymbol{\Omega}_2$ via

$$\underset{k \times (k+p)}{\boldsymbol{\Omega}_1} = \underset{k \times n}{\mathbf{V}_1^*} \; \underset{n \times (k+p)}{\boldsymbol{\Omega}} \quad \text{and} \quad \underset{(n-k) \times (k+p)}{\boldsymbol{\Omega}_2} = \underset{(n-k) \times n}{\mathbf{V}_2^*} \; \underset{n \times (k+p)}{\boldsymbol{\Omega}}.$$

Theorem: [HMT2009,HMT2011] Assuming that $\boldsymbol{\Omega}_1$ is not singular, it holds that

$$|||\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}|||^2 \leq \underbrace{|||\mathbf{D}_2|||^2}_{\text{theoretically minimal error}} + |||\mathbf{D}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}|||^2.$$

Here, $|||\cdot|||$ represents either ℓ^2-operator norm, or the Frobenius norm.

Note: A similar result appears in Boutsidis, Mahoney, Drineas (2009).

Recall: $\mathbf{A} = \mathbf{U}\begin{bmatrix}\mathbf{D}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_2\end{bmatrix}\begin{bmatrix}\mathbf{V}_1^*\\ \mathbf{V}_2^*\end{bmatrix}$, $\begin{bmatrix}\Omega_1\\ \Omega_2\end{bmatrix} = \begin{bmatrix}\mathbf{V}_1^*\Omega\\ \mathbf{V}_2^*\Omega\end{bmatrix}$, $\mathbf{Y} = \mathbf{A}\Omega$, $\mathbf{P}$ proj$^{\mathrm{n}}$ onto Ran($\mathbf{Y}$).

Thm: Suppose $\mathbf{D}_1\Omega_1$ has full rank. Then $\|\mathbf{A} - \mathbf{PA}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|^2$.

Proof: The problem is rotationally invariant $\Rightarrow$ We can assume $\mathbf{U} = \mathbf{I}$ and so $\mathbf{A} = \mathbf{DV}^*$.

Simple calculation: $\|(\mathbf{I} - \mathbf{P})\mathbf{A}\|^2 = \|\mathbf{A}^*(\mathbf{I} - \mathbf{P})^2\mathbf{A}\| = \|\mathbf{D}(\mathbf{I} - \mathbf{P})\mathbf{D}\|$.

$$\mathrm{Ran}(\mathbf{Y}) = \mathrm{Ran}\left(\begin{bmatrix}\mathbf{D}_1\Omega_1\\ \mathbf{D}_2\Omega_2\end{bmatrix}\right) = \mathrm{Ran}\left(\begin{bmatrix}\mathbf{I}\\ \mathbf{D}_2\Omega_2\Omega_1^\dagger\mathbf{D}_1^{-1}\end{bmatrix}\mathbf{D}_1\Omega_1\right) = \mathrm{Ran}\left(\begin{bmatrix}\mathbf{I}\\ \mathbf{D}_2\Omega_2\Omega_1^\dagger\mathbf{D}_1^{-1}\end{bmatrix}\right)$$

Set $\mathbf{F} = \mathbf{D}_2\Omega_2\Omega_1^\dagger\mathbf{D}_1^{-1}$. Then $\mathbf{P} = \begin{bmatrix}\mathbf{I}\\ \mathbf{F}\end{bmatrix}(\mathbf{I} + \mathbf{F}^*\mathbf{F})^{-1}[\mathbf{I}\ \mathbf{F}^*]$. (Compare to $\mathbf{P}_{\mathrm{ideal}} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$.)

Use properties of psd matrices: $\mathbf{I} - \mathbf{P} \preccurlyeq \cdots \preccurlyeq \begin{bmatrix}\mathbf{F}^*\mathbf{F} & -(\mathbf{I} + \mathbf{F}^*\mathbf{F})^{-1}\mathbf{F}^*\\ -\mathbf{F}(\mathbf{I} + \mathbf{F}^*\mathbf{F})^{-1} & \mathbf{I}\end{bmatrix}$

Conjugate by $\mathbf{D}$ to get $\mathbf{D}(\mathbf{I} - \mathbf{P})\mathbf{D} \preccurlyeq \begin{bmatrix}\mathbf{D}_1\mathbf{F}^*\mathbf{F}\mathbf{D}_1 & -\mathbf{D}_1(\mathbf{I} + \mathbf{F}^*\mathbf{F})^{-1}\mathbf{F}^*\mathbf{D}_2\\ -\mathbf{D}_2\mathbf{F}(\mathbf{I} + \mathbf{F}^*\mathbf{F})^{-1}\mathbf{D}_1 & \mathbf{D}_2^2\end{bmatrix}$

Diagonal dominance: $\|\mathbf{D}(\mathbf{I} - \mathbf{P})\mathbf{D}\| \leq \|\mathbf{D}_1\mathbf{F}^*\mathbf{F}\mathbf{D}_1\| + \|\mathbf{D}_2^2\| = \|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|^2 + \|\mathbf{D}_2\|^2$.

Part 2 (out of 3) — bound on expectation of error when Ω is Gaussian:

Let **A** denote an $m \times n$ matrix with singular values $\{\sigma_j\}_{j=1}^{\min(m,n)}$.
Let k denote a target rank and let p denote an over-sampling parameter. Set $\ell = k + p$.
Let Ω denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{A}\Omega)$.

Recall: $|||\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}|||^2 \leq |||\mathbf{D}_2|||^2 + |||\mathbf{D}_2\Omega_2\Omega_1^\dagger|||^2$, where $\Omega_1 = \mathbf{V}_1^*\,\Omega$ and $\Omega_2 = \mathbf{V}_2^*\,\Omega$.

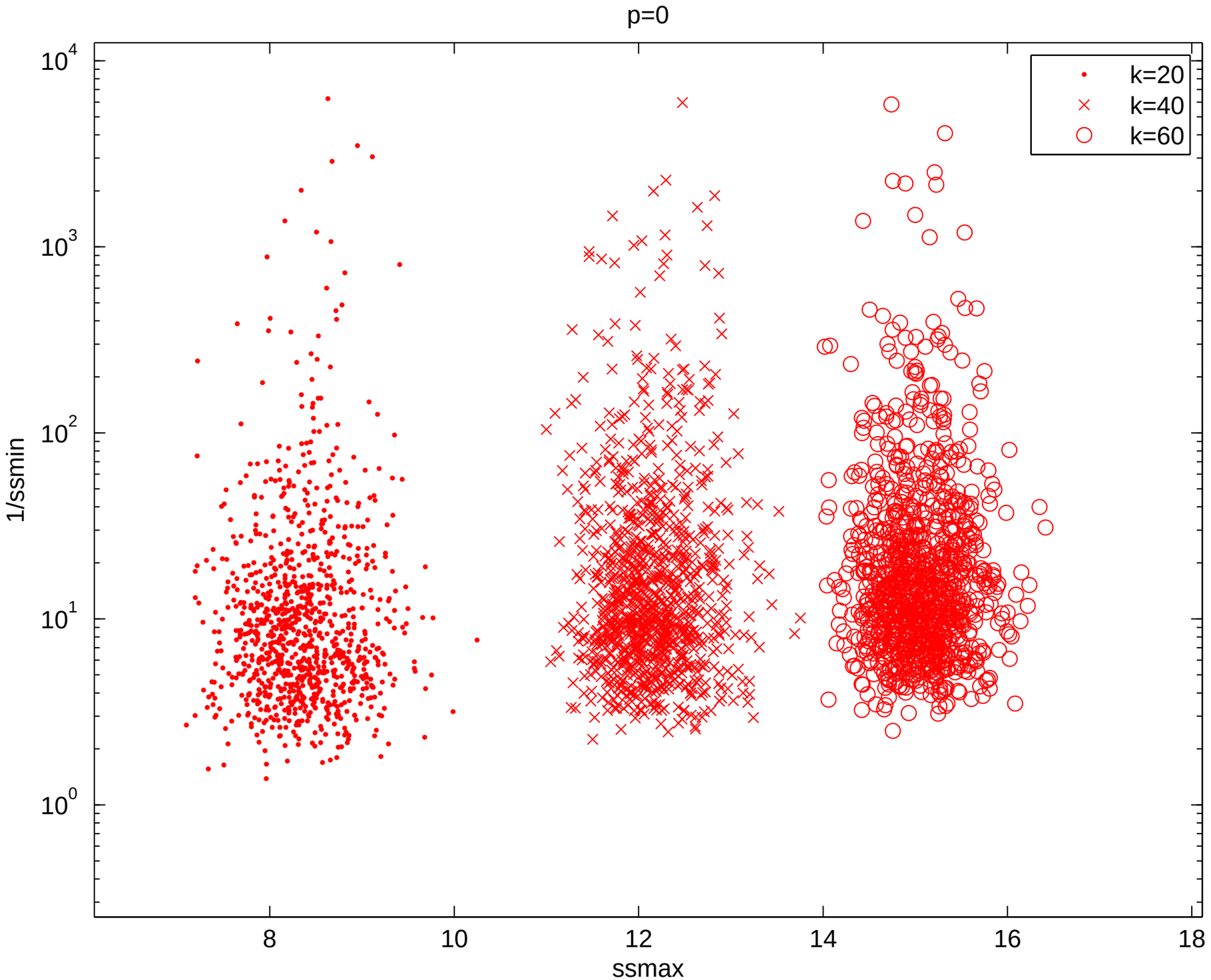
Assumption: Ω is drawn from a normal Gaussian distribution.

Since the Gaussian distribution is rotationally invariant, the matrices Ω_1 and Ω_2 also have a Gaussian distribution. (As a consequence, the matrices **U** and **V** do not enter the analysis and one could simply assume that **A** is diagonal, $\mathbf{A} = \mathrm{diag}(\sigma_1, \sigma_2, \dots)$.)

What is the distribution of $\Omega_1^\dagger$ when Ω_1 is a $k \times (k+p)$ Gaussian matrix?

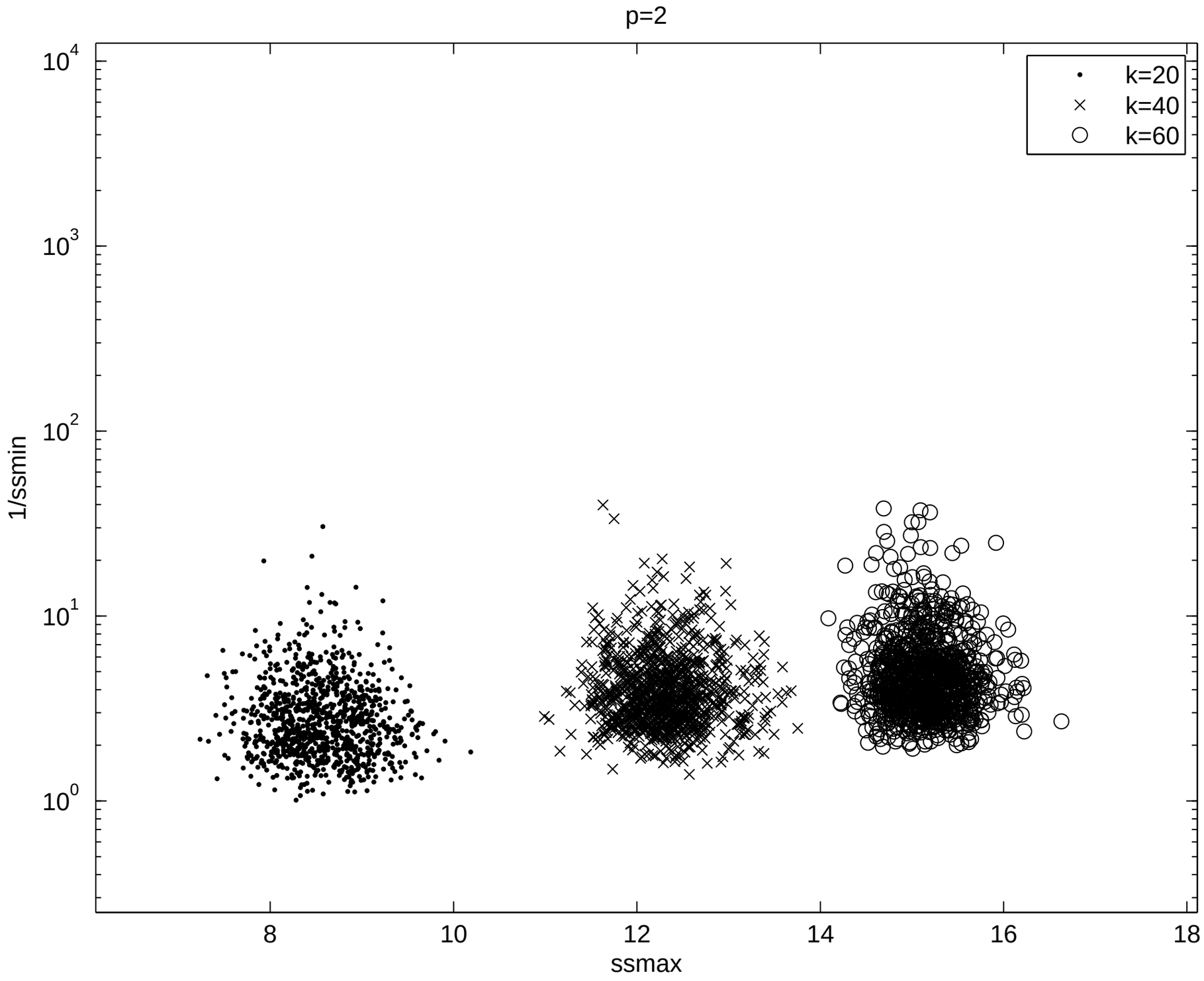
If $p = 0$, then $\|\Omega_1^\dagger\|$ is typically large, and is very unstable.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 0$



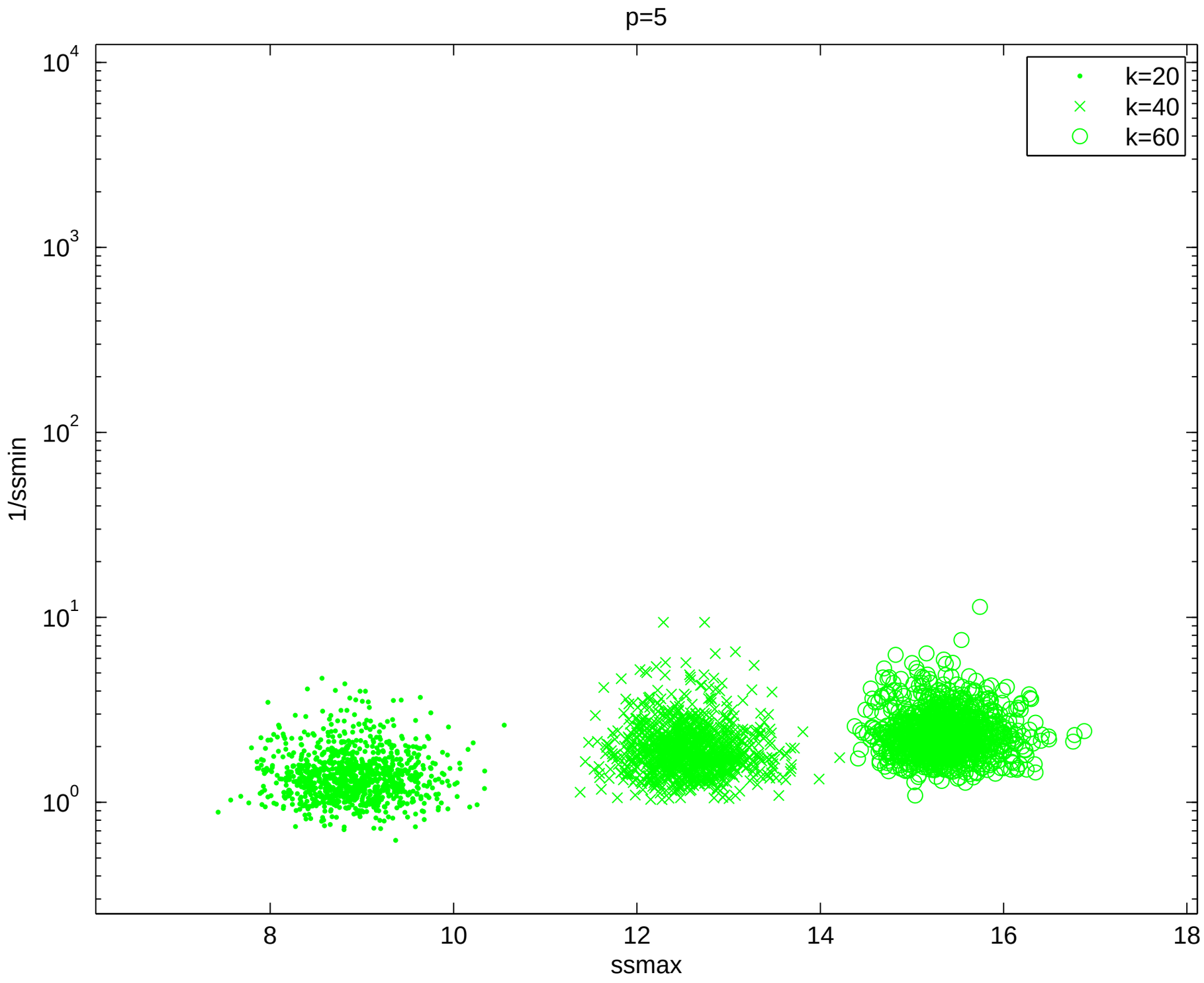
$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 2$



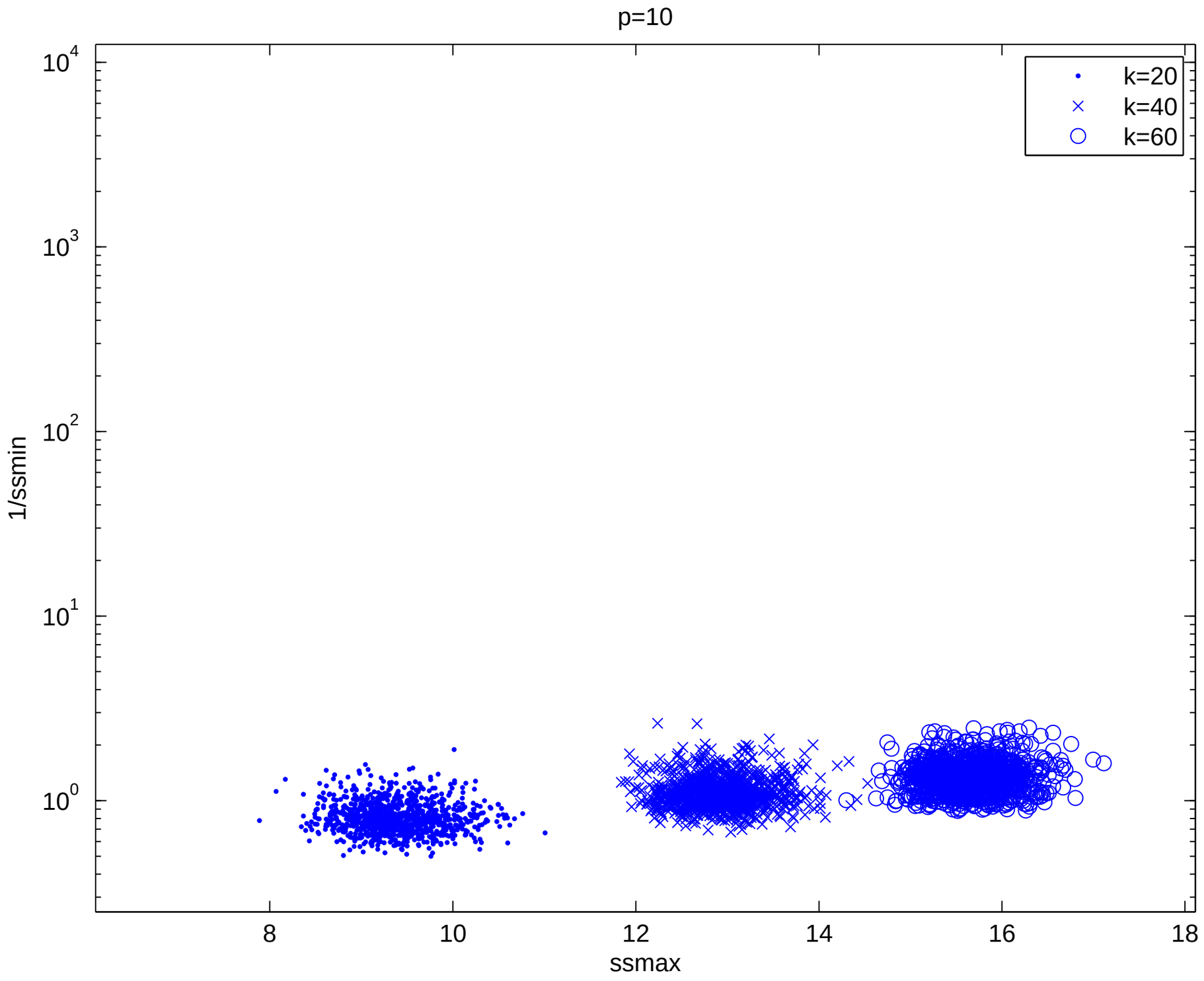
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 5$



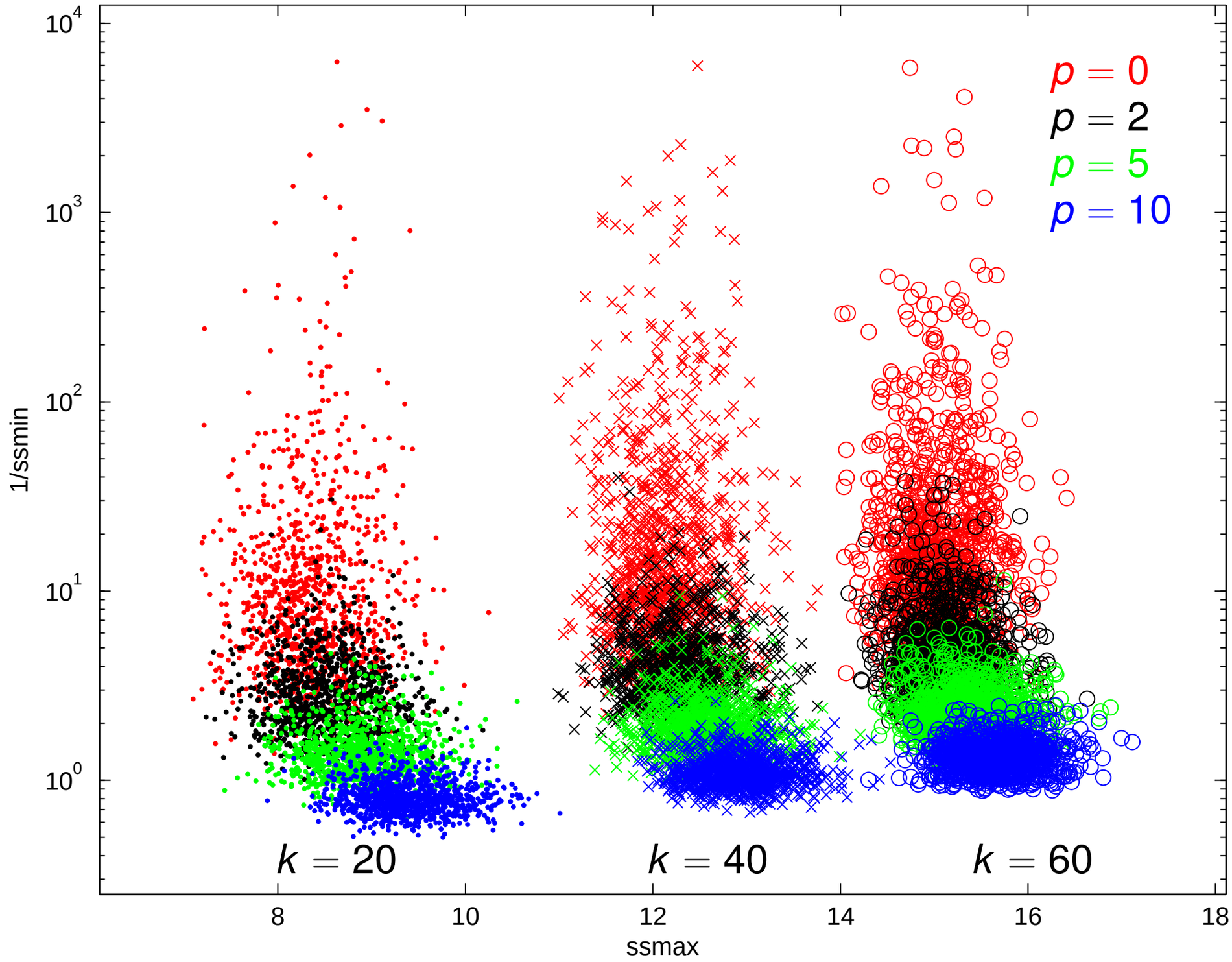
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

Scatter plot showing distribution of $1/\sigma_{\min}$ for $k \times (k+p)$ Gaussian matrices. $p = 10$



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

Scatter plot showing distribution of $k \times (k + p)$ Gaussian matrices.



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

Simplistic proof that a rectangular Gaussian matrix is well-conditioned:

Let **G** denote a $k \times \ell$ Gaussian matrix where $k < \ell$. Let "g" denote a generic $\mathcal{N}(0,1)$ variable and let "r_j" denote a generic random variable distributed like the square root of a χ^2_j variable. Then

$$\mathbf{G} \sim \begin{bmatrix} g & g & g & g & g & g & \cdots \\ g & g & g & g & g & g & \cdots \\ g & g & g & g & g & g & \cdots \\ g & g & g & g & g & g & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \end{bmatrix} \sim \begin{bmatrix} r_\ell & 0 & 0 & 0 & 0 & 0 & \cdots \\ g & g & g & g & g & g & \cdots \\ g & g & g & g & g & g & \cdots \\ g & g & g & g & g & g & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \end{bmatrix}$$

$$\sim \begin{bmatrix} r_\ell & 0 & 0 & 0 & 0 & 0 & \cdots \\ r_{k-1} & g & g & g & g & g & \cdots \\ 0 & g & g & g & g & g & \cdots \\ 0 & g & g & g & g & g & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \end{bmatrix} \sim \begin{bmatrix} r_\ell & 0 & 0 & 0 & 0 & \cdots \\ r_{k-1} & r_{\ell-1} & 0 & 0 & 0 & \cdots \\ 0 & g & g & g & g & \cdots \\ 0 & g & g & g & g & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix}$$

$$\sim \begin{bmatrix} r_\ell & 0 & 0 & 0 & 0 & \cdots \\ r_{k-1} & r_{\ell-1} & 0 & 0 & 0 & \cdots \\ 0 & r_{k-2} & g & g & g & \cdots \\ 0 & 0 & g & g & g & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix} \sim \cdots \sim \begin{bmatrix} r_\ell & 0 & 0 & 0 & 0 & \cdots \\ r_{k-1} & r_{\ell-1} & 0 & 0 & 0 & \cdots \\ 0 & r_{k-2} & r_{\ell-2} & 0 & 0 & \cdots \\ 0 & 0 & r_{k-3} & r_{\ell-3} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix}$$

Gershgorin's circle theorem will now show that **G** is highly likely to be well-conditioned if, e.g., $\ell = 2k$. More sophisticated methods are required to get to $\ell = k + 2$.

Some results on Gaussian matrices. *Adapted from HMT 2009/2011; Gordon (1985,1988) for Proposition 1; Chen & Dongarra (2005) for Propositions 2 and 4; Bogdanov (1998) for Proposition 3.*

Proposition 1: Let $\mathbf{G}$ be a Gaussian matrix. Then

$$\begin{aligned}\left(\mathbb{E}\left[\|\mathbf{SGT}\|_{\mathrm{F}}^2\right]\right)^{1/2} &\leq \|\mathbf{S}\|_{\mathrm{F}}\,\|\mathbf{T}\|_{\mathrm{F}} \\ \mathbb{E}\left[\|\mathbf{SGT}\|\right] &\leq \|\mathbf{S}\|\,\|\mathbf{T}\|_{\mathrm{F}} + \|\mathbf{S}\|_{\mathrm{F}}\,\|\mathbf{T}\|\end{aligned}$$

Proposition 2: Let $\mathbf{G}$ be a Gaussian matrix of size $k \times k+p$ where $p \geq 2$. Then

$$\begin{aligned}\left(\mathbb{E}\left[\|\mathbf{G}^{\dagger}\|_{\mathrm{F}}^2\right]\right)^{1/2} &\leq \sqrt{\frac{k}{p-1}} \\ \mathbb{E}\left[\|\mathbf{G}^{\dagger}\|\right] &\leq \frac{e\sqrt{k+p}}{p}.\end{aligned}$$

Proposition 3: Suppose h is Lipschitz $|h(\mathbf{X}) - h(\mathbf{Y})| \leq L\|\mathbf{X}-\mathbf{Y}\|_{\mathrm{F}}$ and $\mathbf{G}$ is Gaussian. Then

$$\mathbb{P}\left[h(\mathbf{G}) > \mathbb{E}[h(\mathbf{G})] + L\,u\right] \leq e^{-u^2/2}.$$

Proposition 4: Suppose $\mathbf{G}$ is Gaussian of size $k \times k+p$ with $p \geq 4$. Then for $t \geq 1$:

$$\begin{aligned}\mathbb{P}\left[\|\mathbf{G}^{\dagger}\|_{\mathrm{F}} \geq \sqrt{\frac{3k}{p+1}}t\right] &\leq t^{-p} \\ \mathbb{P}\left[\|\mathbf{G}^{\dagger}\| \geq \frac{e\sqrt{k+p}}{p+1}t\right] &\leq t^{-(p+1)}\end{aligned}$$

Recall: $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|^2$, where Ω_1 and Ω_2 are Gaussian and Ω_1 is $k \times k + p$.

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

Proof: First observe that

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| = \mathbb{E}\big(\|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|^2\big)^{1/2} \leq \|\mathbf{D}_2\| + \mathbb{E}\|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|.$$

Condition on Ω_1 and use Proposition 1:

$$\begin{aligned}\mathbb{E}\|\mathbf{D}_2\Omega_2\Omega_1^\dagger\| &\leq \mathbb{E}\big[\|\mathbf{D}_2\|\,\|\Omega_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\Omega_1^\dagger\|\big] \\ &\leq \{\text{Hölder}\} \leq \|\mathbf{D}_2\|\left(\mathbb{E}\|\Omega_1^\dagger\|_{\mathrm{F}}^2\right)^{1/2} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\mathbb{E}\|\Omega_1^\dagger\|.\end{aligned}$$

Proposition 2 now provides bounds for $\mathbb{E}\|\Omega_1^\dagger\|_{\mathrm{F}}^2$ and $\mathbb{E}\|\Omega_1^\dagger\|$ and we get

$$\mathbb{E}\|\mathbf{D}_2\Omega_2\Omega_1^\dagger\| \leq \sqrt{\frac{k}{p-1}}\|\mathbf{D}_2\| + \frac{e\sqrt{k+p}}{p}\|\mathbf{D}_2\|_{\mathrm{F}} = \sqrt{\frac{k}{p-1}}\sigma_{k+1} + \frac{e\sqrt{k+p}}{p}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}.$$

Some results on Gaussian matrices. *Adapted from HMT2009/2011; Gordon (1985,1988) for Proposition 1; Chen & Dongarra (2005) for Propositions 2 and 4; Bogdanov (1998) for Proposition 3.*

Proposition 1: Let $\mathbf{G}$ be a Gaussian matrix. Then

$$\begin{aligned}\left(\mathbb{E}\left[\|\mathbf{SGT}\|_{\mathrm{F}}^2\right]\right)^{1/2} &\leq \|\mathbf{S}\|_{\mathrm{F}}\,\|\mathbf{T}\|_{\mathrm{F}} \\ \mathbb{E}\left[\|\mathbf{SGT}\|\right] &\leq \|\mathbf{S}\|\,\|\mathbf{T}\|_{\mathrm{F}} + \|\mathbf{S}\|_{\mathrm{F}}\,\|\mathbf{T}\|\end{aligned}$$

Proposition 2: Let $\mathbf{G}$ be a Gaussian matrix of size $k \times k+p$ where $p \geq 2$. Then

$$\begin{aligned}\left(\mathbb{E}\left[\|\mathbf{G}^\dagger\|_{\mathrm{F}}^2\right]\right)^{1/2} &\leq \sqrt{\frac{k}{p-1}} \\ \mathbb{E}\left[\|\mathbf{G}^\dagger\|\right] &\leq \frac{e\sqrt{k+p}}{p}.\end{aligned}$$

Proposition 3: Suppose h is Lipschitz $|h(\mathbf{X}) - h(\mathbf{Y})| \leq L\|\mathbf{X}-\mathbf{Y}\|_{\mathrm{F}}$ and $\mathbf{G}$ is Gaussian. Then

$$\mathbb{P}\left[h(\mathbf{G}) > \mathbb{E}[h(\mathbf{G})] + L\,u\right] \leq e^{-u^2/2}.$$

Proposition 4: Suppose $\mathbf{G}$ is Gaussian of size $k \times k+p$ with $p \geq 4$. Then for $t \geq 1$:

$$\begin{aligned}\mathbb{P}\left[\|\mathbf{G}^\dagger\|_{\mathrm{F}} \geq \sqrt{\frac{3k}{p+1}}t\right] &\leq t^{-p} \\ \mathbb{P}\left[\|\mathbf{G}^\dagger\| \geq \frac{e\sqrt{k+p}}{p+1}t\right] &\leq t^{-(p+1)}\end{aligned}$$

Recall: $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|^2$, where Ω_1 and Ω_2 are Gaussian and Ω_1 is $k \times k+p$.

Theorem: With probability at least $1 - 2\,t^{-p} - e^{-u^2/2}$ it holds that

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \leq \left(1 + t\sqrt{\frac{3k}{p+1}} + u\,t\,\frac{e\,\sqrt{k+p}}{p+1}\right)\,\sigma_{k+1} + \frac{t\,e\,\sqrt{k+p}}{p+1}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}.$$

Proof: Set $E_t = \left\{\|\Omega_1\| \leq \frac{e\sqrt{k+p}}{p+1}t \quad \text{and} \quad \|\Omega_1^\dagger\|_{\mathrm{F}} \leq \sqrt{\frac{3k}{p+1}}\,t\right\}$. By Proposition 4: $\mathbb{P}(E_t^{\mathrm{c}}) \leq 2\,t^{-p}$.

Set $h(\mathbf{X}) = \|\mathbf{D}_2\mathbf{X}\Omega_1^\dagger\|$. A direct calculation shows

$$|h(\mathbf{X}) - h(\mathbf{Y})| \leq \|\mathbf{D}_2\|\,\|\Omega_1^\dagger\|\,\|\mathbf{X} - \mathbf{y}\|_{\mathrm{F}}.$$

Hold Ω_1 fixed and take the expectation on Ω_2. Then Proposition 1 applies and so

$$\mathbb{E}\big[h(\Omega_2) \;\big|\; \Omega_1\big] \leq \|\mathbf{D}_2\|\,\|\Omega_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\Omega_1^\dagger\|.$$

Now use Proposition 3 (concentration of measure)

$$\mathbb{P}\Big[\underbrace{\|\mathbf{D}_2\Omega_2\Omega_1^\dagger\|}_{=h(\Omega_2)} > \underbrace{\|\mathbf{D}_2\|\,\|\Omega_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\Omega_1^\dagger\|}_{=\mathbb{E}[h(\Omega_2)]} + \underbrace{\|\mathbf{D}_2\|\,\|\Omega_1^\dagger\|}_{=L}\,u \;\Big|\; E_t\Big] < e^{-u^2/2}.$$

When E_t holds true, we have bounds on the "badness" of $\Omega_1^\dagger$:

$$\mathbb{P}\left[\|\mathbf{D}_2\Omega_2\Omega_1^\dagger\| > \|\mathbf{D}_2\|\sqrt{\frac{3k}{p+1}}t + \|\mathbf{D}_2\|_{\mathrm{F}}\,\frac{e\sqrt{k+p}}{p+1}t + \|\mathbf{D}_2\|\,\frac{e\sqrt{k+p}}{p+1}\,ut \;\middle|\; E_t\right] < e^{-u^2/2}.$$

The theorem is obtained by using $\mathbb{P}(E_t^{\mathrm{c}}) \leq 2\,t^{-p}$ to remove the conditioning of E_t.

Concluding remarks:

- Techniques based on random embeddings have proven to be very effective for solving large scale linear algebraic problems.

- The approximation error is a random variable, but its distribution is tightly concentrated. Rigorous error bounds that are satisfied with probability $1 - \eta$ where η is a user set "failure probability" (*e.g.* $\eta = 10^{-10}$).

- Techniques based on *sampling* enable certain otherwise impossible computations. However, these techniques behave like Monte Carlo methods, with errors converging as $\varepsilon \sim 1/\sqrt{n}$, so that $n \sim 1/\varepsilon^2$ samples are required.

- Reducing *communication* has been a recurring theme.

- These lectures mentioned *error estimators* only briefly, but they are important.
  Can operate independently of the algorithm for improved robustness.
  Typically cheap and easy to implement. Used to determine the actual rank.

- The theory can be hard (at least for me), but *experimentation is easy!*
  Concentration of measure makes the algorithms behave as if deterministic.

Slides and links: `http://users.oden.utexas.edu/~pgm/2020_kth_course/`

Surveys:

- 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.

- 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 rather than theory.

- 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.

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)