Randomized methods for accelerating matrix factorization algorithms

Gunnar Martinsson

The University of Oxford

Collaborators: Edo Liberty, Gregorio Quintana-Ortí, Vladimir Rokhlin, Yoel Shkolnisky, Arthur Szlam, Joel Tropp, Mark Tygert, Robert van de Geijn, Franco Woolfe, ...

Students and postdoc: Nathan Halko, Nathan Heavner, Sergey Voronin.

Papers, software, etc: `http://people.maths.ox.ac.uk/martinsson/`

Research support by:

Objective:

Given an $m \times n$ matrix $\mathbf{A}$, we seek to compute a rank-k approximation,

typically with $k \ll \min(m, n)$,

$$\mathbf{A} \approx \mathbf{E} \mathbf{F}^* = \sum_{j=1}^{k} \mathbf{e}_j \mathbf{f}_j^*.$$

$$m \times n \quad m \times k \;\; k \times n$$

Solving this problem leads to algorithms for computing:

- Eigenvectors corresponding to leading eigenvalues.
  (Require $\mathbf{e}_j = \lambda_j \mathbf{f}_j$, and $\{\mathbf{f}_j\}_{j=1}^k$ to be orthonormal.)
- Singular Value Decomposition (SVD) / Principal Component Analysis (PCA).
  (Require $\{\mathbf{e}_j\}_{j=1}^k$ and $\{\mathbf{f}_j\}_{j=1}^k$ to be orthogonal sets.)
- Spanning columns or rows.
  (Require $\{\mathbf{e}_j\}_{j=1}^k$ to be columns of $\mathbf{A}$, or require $\{\mathbf{f}_j^*\}_{j=1}^k$ to be rows of $\mathbf{A}$.)
- Etc.

Later in the talk, we will also discuss techniques for full (exact) factorizations.

Applications:

- Accelerating standard packages for linear algebra.
- Fast algorithms for elliptic PDEs: more efficient Fast Multipole Methods, fast *direct* solvers, construction of special quadratures for corners and edges, etc.
- Statistical analysis of large data sets. E.g. via Principal Component Analysis (PCA).
- Data mining (machine learning, analysis of network matrices, image processing, etc). Beside problems that are immediately formulated as matrix computations, the techniques described can accelerate problems like nearest neighbor search for large clouds of points in high dimensional space, clustering, etc.
- Preconditioners for solving linear systems $\mathbf{Ax} = \mathbf{b}$ applicable to broad classes of matrices. (General matrices?)
- Diffusion geometry; a technique for constructing parameterizations on large collections of data points organized (modulo noise) along non-linear low-dimensional manifolds. Requires the computations of eigenvectors of *graph Laplace operators*.
- Etc.

Review of existing methods I

For a dense $n \times n$ matrix that fits in RAM, excellent algorithms are well known.

Foundation of software packages such as LAPACK, Intel MKL, Matlab, *etc*).

- Double precision accuracy.
- Very stable.
- $O(n^3)$ asymptotic complexity. Reasonably small constants.
- Require extensive "random" access to the matrix. $\rightarrow$ *Hard to parallelize.*

When the target rank k is much smaller than n, there also exist $O(n^2\,k)$ methods with similar characteristics (the well-known Golub-Businger method, RRQR by Gu and Eisentstat, *etc*).

For small matrices, the state-of-the-art is somewhat satisfactory.

(By "small," we mean something like $n \leq 10\,000$ on today's computers.)

Review of existing methods I

For a dense $n \times n$ matrix that fits in RAM, excellent algorithms are well known.

Foundation of software packages such as LAPACK, Intel MKL, Matlab, *etc*).

- Double precision accuracy.
- Very stable.
- $O(n^3)$ asymptotic complexity. Reasonably small constants.
- Require extensive "random" access to the matrix. $\rightarrow$ *Hard to parallelize.*

When the target rank k is much smaller than n, there also exist $O(n^2\,k)$ methods with similar characteristics (the well-known Golub-Businger method, RRQR by Gu and Eisentstat, *etc*).

For small matrices, the state-of-the-art is somewhat satisfactory.

(By "small," we mean something like $n \leq 10\,000$ on today's computers.)

Well-established field, but it turns out dramatic improvements are possible.

Review of existing methods II

If the matrix is large, but can rapidly be applied to a vector (if it is sparse, or sparse in Fourier space, or amenable to the FMM, etc.), so called *Krylov subspace methods* often yield excellent accuracy and speed.

The idea is to 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.

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.).

"New" challenges in algorithmic design:

The existing state-of-the-art methods of numerical linear algebra that we have very briefly outlined were designed for an environment where the matrix fits in RAM and the key to performance was to minimize the number of *floating point operations* required.

Currently, *communication* is becoming the real bottleneck:

- While clock speed is hardly improving at all anymore, the cost of a flop keeps going down rapidly. (Multi-core processors, GPUs, cloud computing, etc.)

- The cost of slow storage (hard drives, flash memory, etc.) is also going down rapidly.

- Communication costs are decreasing, but *not* rapidly.
  - Moving data from a hard-drive.
  - Moving data between nodes of a parallel machine. (Or cloud computer ...)
  - The amount of fast cache memory close to a processor is not improving much.
    (In fact, it could be said to be *shrinking* — GPUs, multi-core, etc.)

- "Deluge of data". Driven by ever cheaper storage and acquisition techniques. Web search, data mining in archives of documents or photos, hyper-spectral imagery, social networks, gene arrays, proteomics data, sensor networks, financial transactions, ...

The more powerful computing machinery becomes,
the more important efficient algorithm design becomes.

- Linear scaling (w.r.t. problem size, processors, etc.).
- Minimal data movement.

Review of existing methods III — randomization (early results)

That *randomization* can be used to overcome some of the communication bottlenecks in matrix computations has been pointed out by several authors:

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala (2000)

A. Frieze, R. Kannan, and S. Vempala (1999, 2004)

D. Achlioptas and F. McSherry (2001)

P. Drineas, R. Kannan, M. W. Mahoney, and S. Muthukrishnan (2006)

S. Har-Peled (2006)

A. Deshpande and S. Vempala (2006)

S. Friedland, M. Kaveh, A. Niknejad, and H. Zare (2006)

T. Sarlós (2006a, 2006b, 2006c)

M. Rudelson, R. Vershynin (2007)

K. Clarkson, D. Woodruff (2009)

. . . deluge of papers . . .

Literature surveys: Halko, Martinsson, Tropp (2011). Mahoney (2011). Woodruff (2014). Etc.

Review of existing methods III — randomization

Examples of how randomization could be used:

- **Random column/row selection**
  Draw at random some columns and suppose that they span the entire column space.
  If rows are drawn as well, then spectral properties can be estimated.
  Crude sampling leads to less than $O(mn)$ complexity, but is very dangerous.

- **Sparsification**
  Zero out the vast majority of the entries of the matrix. Keep a random subset of entries, and boost their magnitude to preserve "something."

- **Quantization and sparsification**
  Restrict the entries of the matrix to a small set of values (-1/0/1 for instance).

The methods outlined can be as fast as you like, but must necessarily have very weak performance guarantees. They can work well for certain classes of matrices for which additional information is available (basically, matrices that are in some sense "over-sampled").

Approach advocated here:

A randomized algorithm for computing a rank-k approximation to an $m \times n$ matrix.

It is engineered from the ground up to:

- Minimize communication.
- Handle streaming data, or data stored "out-of-core."
- Easily adapt to a broad range of distributed computing architectures.

Computational profile:

- At least $O(mn)$ complexity. To be precise: $O(mnk)$ or $O(mn\log(k))$.
- The accuracy ε is a user-set number.

  (If the application permits, it could be as close to ϵ_{mach} as you like.)
- Since the method is randomized, it has a *failure probability* η.

  η is a user specified number.

  The cost of the method grows as $\eta \to 0$, but setting $\eta = 10^{-10}$ is cheap.

  For all practical purposes, the method succeeds with probability 1.

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

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. *Find an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\,\mathbf{Q}^*\,\mathbf{A}$.*
2. *(I.e., the columns of $\mathbf{Q}$ form an ON-basis for the range of $\mathbf{A}$.)*
3.
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
5. Compute the SVD of the small matrix $\mathbf{B}$ so that $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$.
6. Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$.

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. *Find an $m \times k$ orthonormal matrix $\mathbf{Q}$ such that $\mathbf{A} \approx \mathbf{Q}\,\mathbf{Q}^*\,\mathbf{A}$.*
2. *(I.e., the columns of $\mathbf{Q}$ form an ON-basis for the range of $\mathbf{A}$.)*
3.
4. Form the $k \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$.
5. Compute the SVD of the small matrix $\mathbf{B}$ so that $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$.
6. Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$.

Note: Steps 4 – 6 are exact; the error in the method is all in $\mathbf{Q}$:

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

Note: The classical *Golub-Businger algorithm* follows this pattern. It finds $\mathbf{Q}$ in Step 3 via direct orthogonalization of the columns of $\mathbf{A}$ via, e.g., Gram-Schmidt.

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 random vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later — think Gaussian for now.)

2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\,\mathbf{r}_1, \mathbf{y}_2 = \mathbf{A}\,\mathbf{r}_2, \ldots, \mathbf{y}_k = \mathbf{A}\,\mathbf{r}_k \in \mathbb{R}^m$.

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.

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 random vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later — think Gaussian for now.)

2. Form "sample" vectors $\mathbf{y}_1 = \mathbf{A}\,\mathbf{r}_1, \mathbf{y}_2 = \mathbf{A}\,\mathbf{r}_2, \ldots, \mathbf{y}_k = \mathbf{A}\,\mathbf{r}_k \in \mathbb{R}^m$.

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 (which happen to be the k leading left singular vectors).

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 random matrix $\mathbf{R} \in \mathbb{R}^{n \times k}$.
   (We will discuss the choice of distribution later — think Gaussian for now.)

2. Form a "sample" matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{R} \in \mathbb{R}^{m \times k}$.

3. Form an orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{m \times k}$ such that $\mathbf{Y} = \mathbf{QR}$.
   For instance, Gram-Schmidt can be used — pivoting is rarely required.

If $\mathbf{A}$ has *exact* rank k, then $\mathbf{A} = \mathbf{Q}\mathbf{Q}^*\mathbf{A}$ with probability 1.

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 $\mathbf{R}$.
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{R}$.
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 $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$.

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 $\mathbf{R}$. `R = randn(n,k)`
2. Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{A}\,\mathbf{R}$. `Y = A * R`
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`

Single pass algorithms:

A is symmetric:	**A is not symmetric:**
Generate a random matrix $\mathbf{G}$.	Generate random matrices $\mathbf{G}$ and $\mathbf{H}$.
Compute a sample matrix $\mathbf{Y}$.	Compute sample matrices $\mathbf{Y} = \mathbf{A}\,\mathbf{G}$ and $\mathbf{Z} = \mathbf{A}^*\,\mathbf{H}$.
Find an ON matrix $\mathbf{Q}$ such that $\mathbf{Y} = \mathbf{Q}\,\mathbf{Q}^*\,\mathbf{Y}$.	Find ON matrices $\mathbf{Q}$ and $\mathbf{W}$ such that $\mathbf{Y} = \mathbf{Q}\,\mathbf{Q}^*\,\mathbf{Y}$ and $\mathbf{Z} = \mathbf{W}\,\mathbf{W}^*\,\mathbf{Z}$.
Solve for $\mathbf{T}$ the linear system $\mathbf{Q}^*\,\mathbf{Y} = \mathbf{T}\,(\mathbf{Q}^*\,\mathbf{G})$.	Solve for $\mathbf{T}$ the linear systems $\mathbf{Q}^*\,\mathbf{Y} = \mathbf{T}\,(\mathbf{W}^*\,\mathbf{G})$ and $\mathbf{W}^*\,\mathbf{Z} = \mathbf{T}^*\,(\mathbf{Q}^*\,\mathbf{H})$.
Factor $\mathbf{T}$ so that $\mathbf{T} = \hat{\mathbf{U}}\,\mathbf{D}\,\hat{\mathbf{U}}^*$.	Factor $\mathbf{T}$ so that $\mathbf{T} = \hat{\mathbf{U}}\,\mathbf{D}\,\hat{\mathbf{V}}^*$.
Form $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$.	Form $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$ and $\mathbf{V} = \mathbf{W}\,\hat{\mathbf{V}}$.
Output: $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{U}^*$	Output: $\mathbf{A} \approx \mathbf{U}\mathbf{D}\mathbf{V}^*$

Note: With $\mathbf{B}$ as on the previous slide we have $\mathbf{T} \approx \mathbf{B}\,\mathbf{Q}$ (sym. case) and $\mathbf{T} \approx \mathbf{B}\,\mathbf{W}$ (nonsym. case).

References: *Woolfe, Liberty, Rokhlin, and Tygert (2008), Clarkson and Woodruff (2009), Halko, Martinsson and Tropp (2009).*

Cost of randomized methods vs. classical (deterministic) methods:

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 $O(mn\log(k))$ if a structured random matrix is used. For instance, $\mathbf{R}$ can be a sub-sampled randomized Fourier transform, which can be applied rapidly using variations of the FFT.

Cost of randomized methods vs. classical (deterministic) methods:

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 $O(mn\log(k))$ if a structured random matrix is used. For instance, $\mathbf{R}$ can be a sub-sampled randomized Fourier transform, 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 and Chazelle (2006); Liberty, Rokhlin, Tygert, and Woolfe (2006). Halko, Martinsson, Tropp (2011).
Much subsequent work — "Fast Johnson-Lindenstrauss transform."

Cost of randomized methods vs. classical (deterministic) methods:

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 $O(mn\log(k))$ if a structured random matrix is used. For instance, $\mathbf{R}$ can be a sub-sampled randomized Fourier transform, which can be applied rapidly using variations of the FFT.

Cost of randomized methods vs. classical (deterministic) methods:

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 $O(mn\log(k))$ if a structured random matrix is used. For instance, **R** can be a sub-sampled randomized Fourier transform, 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...)

Cost of randomized methods vs. classical (deterministic) methods:

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 $O(mn\log(k))$ if a structured random matrix is used. For instance, **R** can be a sub-sampled randomized Fourier transform, 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.

Practical speed of $O(mn\log(k))$ complexity randomized SVD

Consider the task of computing a rank-k SVD of a matrix $\mathbf{A}$ of size $n \times n$.

$t^{(\mathrm{direct})}$ Time for classical (Golub-Businger) method — $O(k\,n^2)$

$t^{(\mathrm{srft})}$ Time for randomized method with an SRFT — $O(\log(k)\,n^2)$

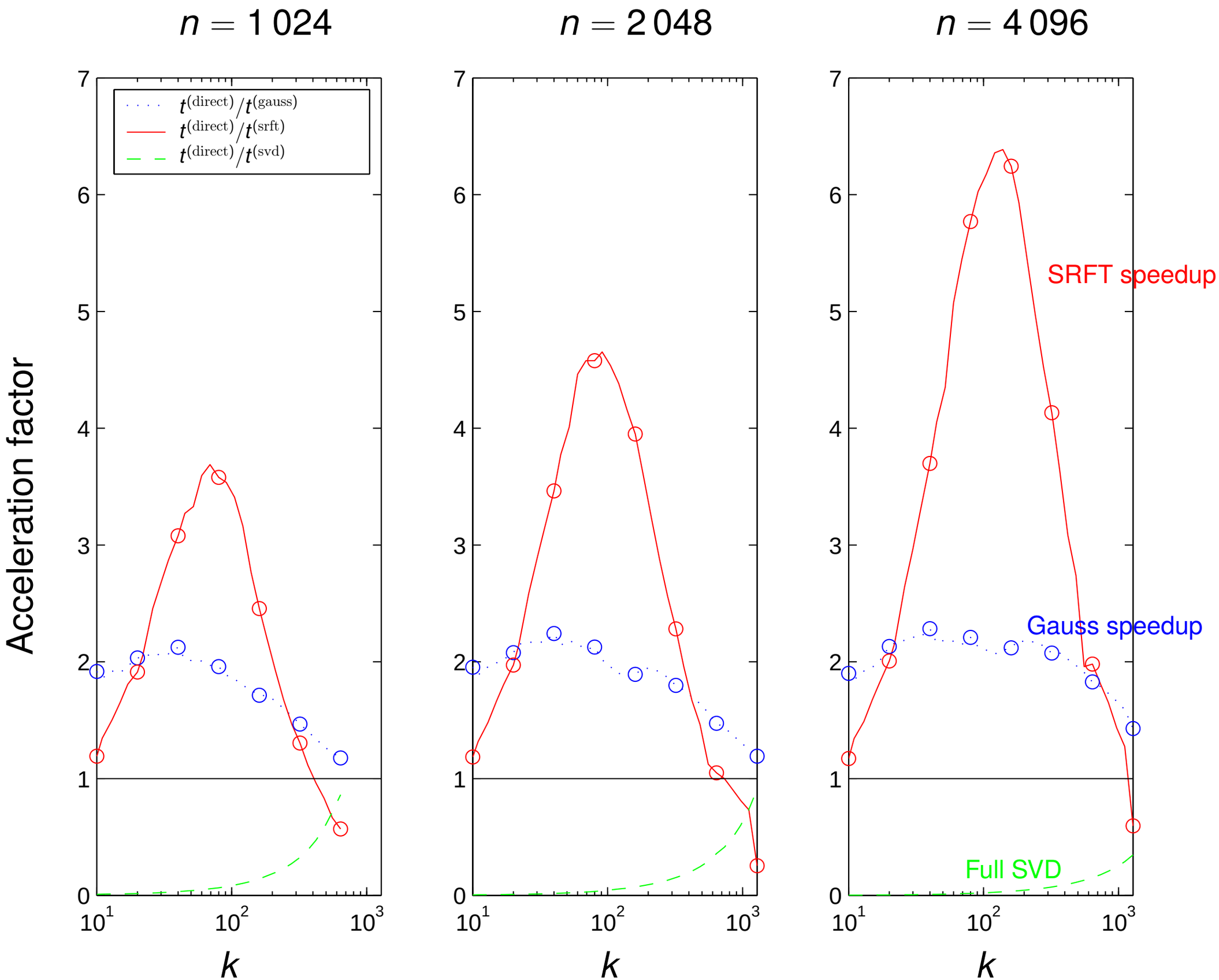
$t^{(\mathrm{gauss})}$ Time for randomized method with a Gaussian matrix — $O(k\,n^2)$

$t^{(\mathrm{svd})}$ Time for a full SVD — $O(n^3)$

We will show the

acceleration factors: $\dfrac{t^{(\mathrm{direct})}}{t^{(\mathrm{srft})}}$ $\dfrac{t^{(\mathrm{direct})}}{t^{(\mathrm{gauss})}}$ $\dfrac{t^{(\mathrm{direct})}}{t^{(\mathrm{svd})}}$

for different values of n and k.



Observe: Large speedups (up to a factor 6!) for moderate size matrices.

Theory

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 $n \times k$ random matrix $\mathbf{R}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $n \times k$ random matrix $\mathbf{R}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $n \times k$ random matrix $\mathbf{R}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $n \times k$ random matrix $\mathbf{R}$.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $n \times k$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times k$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $n \times (k+p)$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times (k+p)$ sample matrix $\mathbf{Y} = \mathbf{AR}$.	(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 $\mathbf{R}$ denote an $n \times (k+p)$ Gaussian matrix.

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

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

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

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

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

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

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 $3\,p^{-p}$.

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 **R** denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AR})$.

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

1. Make no assumption on **R**. Construct a deterministic bound of the form

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

2. Assume that **R** is drawn from a normal Gaussian distribution.
   Take expectations of the deterministic bound to attain a bound of the form

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

3. Assume that **R** is drawn from a normal Gaussian distribution.
   Take expectations of the deterministic bound conditioned on "bad behavior" in **R** to get that

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

   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 **R** denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \texttt{orth}(\mathbf{AR})$.

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

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

Define $\mathbf{R}_1$ and $\mathbf{R}_2$ via

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

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

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

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

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

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

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

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

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

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

Set $\mathbf{F} = \mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\mathbf{D}_1^{-1}$. 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: $\|\mathbf{D}(\mathbf{I} - \mathbf{P})\mathbf{D}\| \leq \|\mathbf{D}_1\mathbf{F}^*\mathbf{F}\mathbf{D}_1\| + \|\mathbf{D}_2^2\| = \|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|^2 + \|\mathbf{D}_2\|^2$.

Part 2 (out of 3) — bound on expectation of error when R 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 **R** denote an $n \times \ell$ "test matrix", and let **Q** denote the $m \times \ell$ matrix $\mathbf{Q} = \mathtt{orth}(\mathbf{AR})$.

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

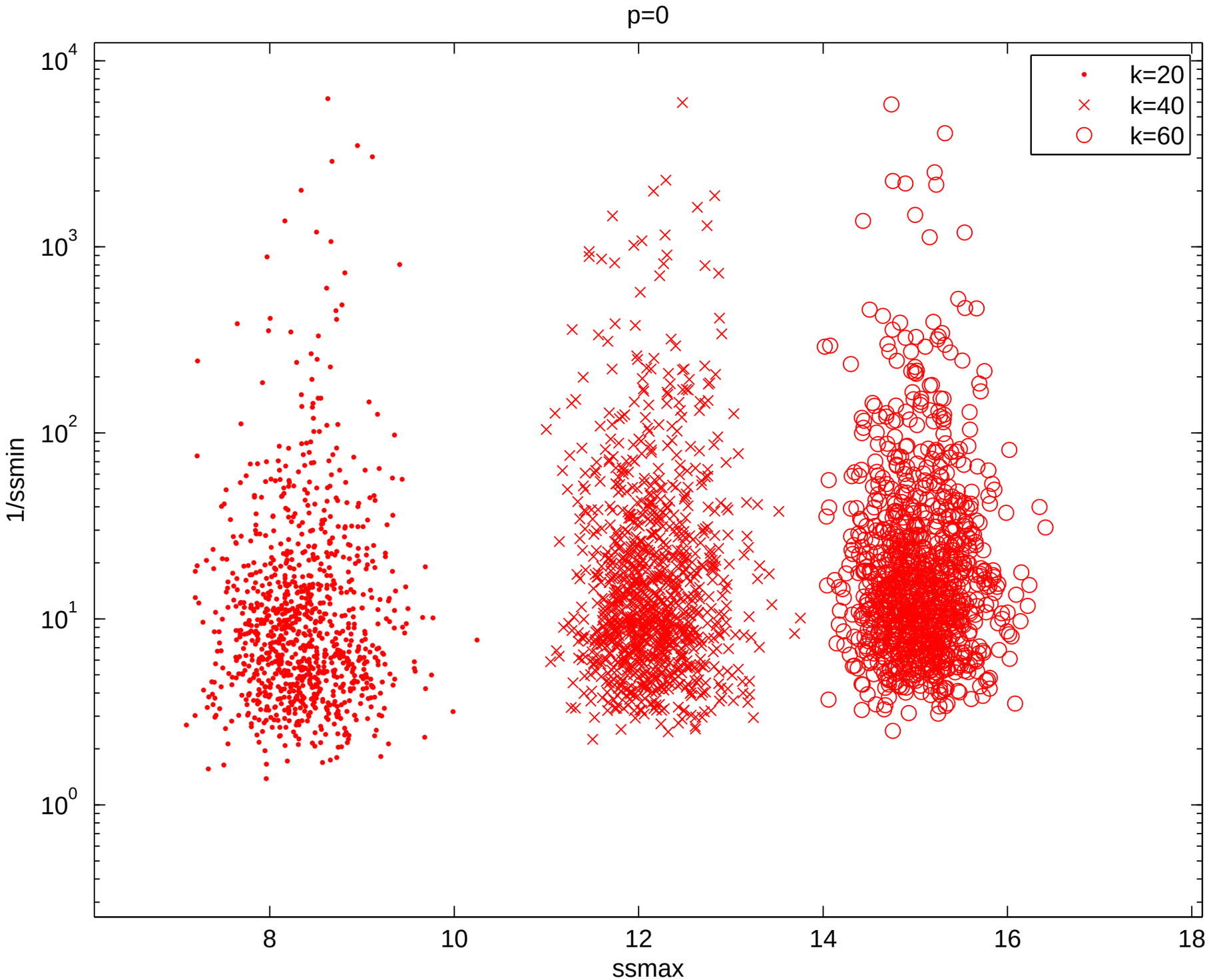
Assumption: **R** is drawn from a normal Gaussian distribution.

Since the Gaussian distribution is rotationally invariant, the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ 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, $\mathbf{A} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots)$.)

What is the distribution of $\mathbf{R}_1^\dagger$ when $\mathbf{R}_1$ is a $k \times (k+p)$ Gaussian matrix?

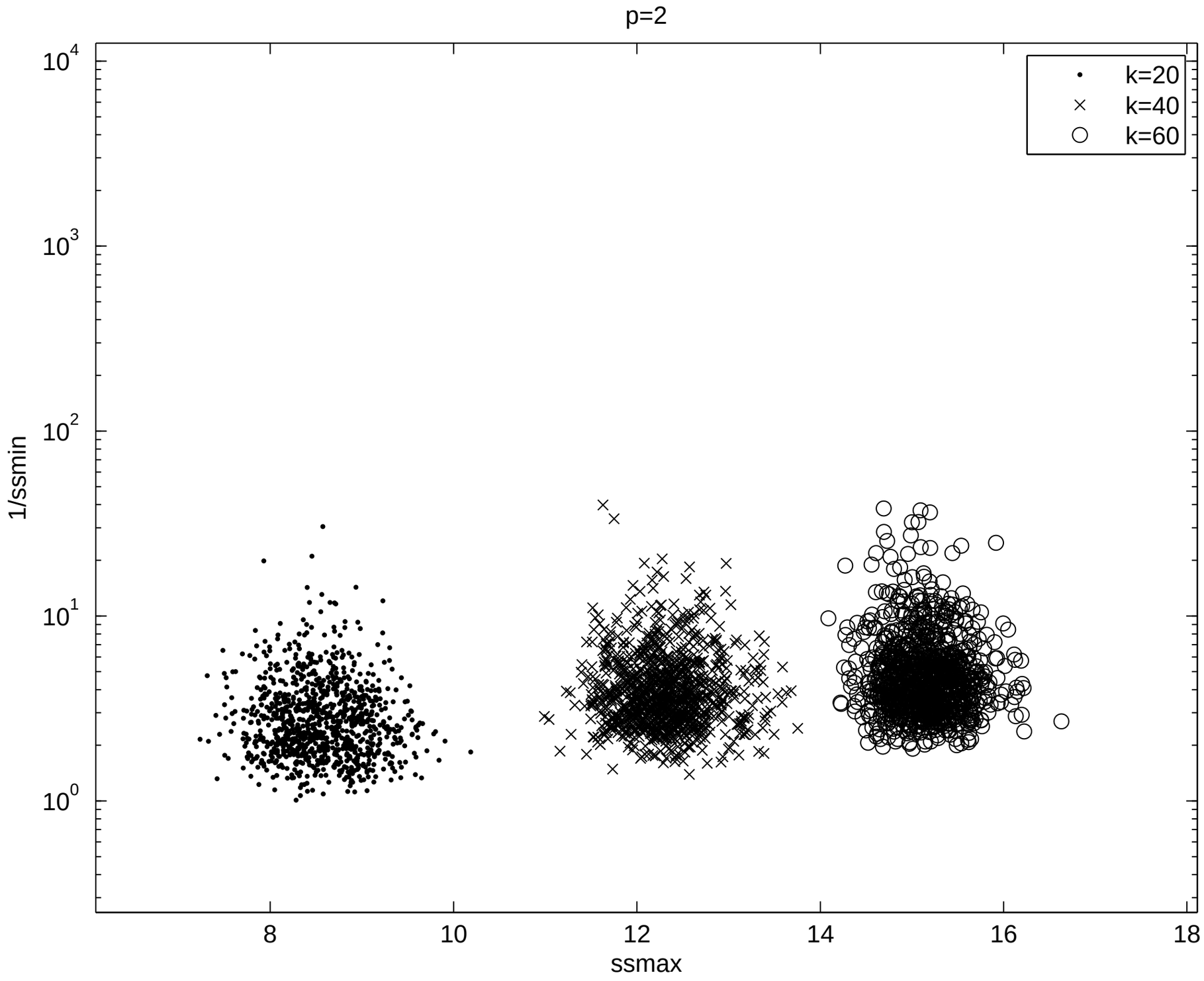
If $p = 0$, then $\|\mathbf{R}_1^\dagger\|$ is typically large, and is very unstable.

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



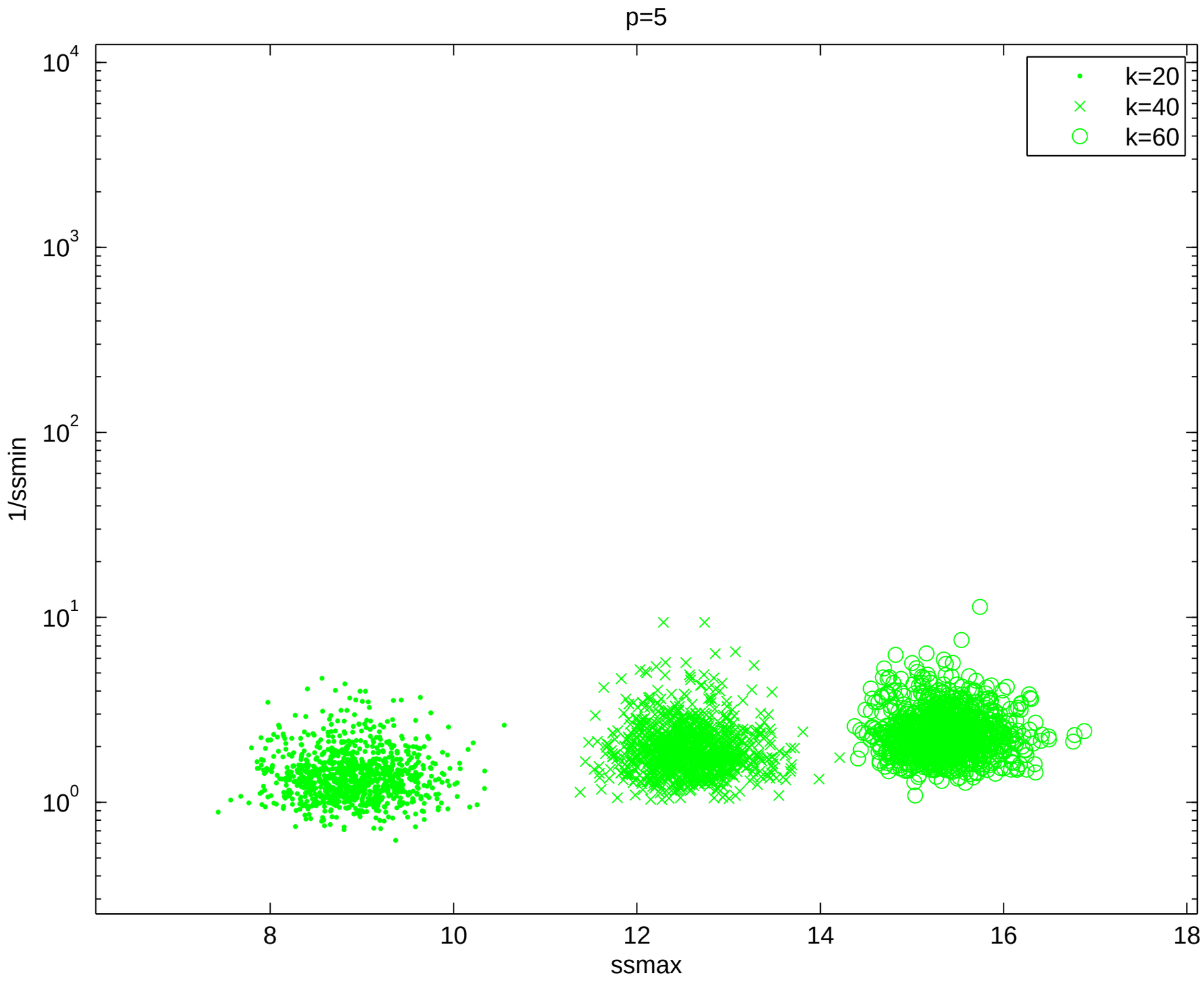
$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

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



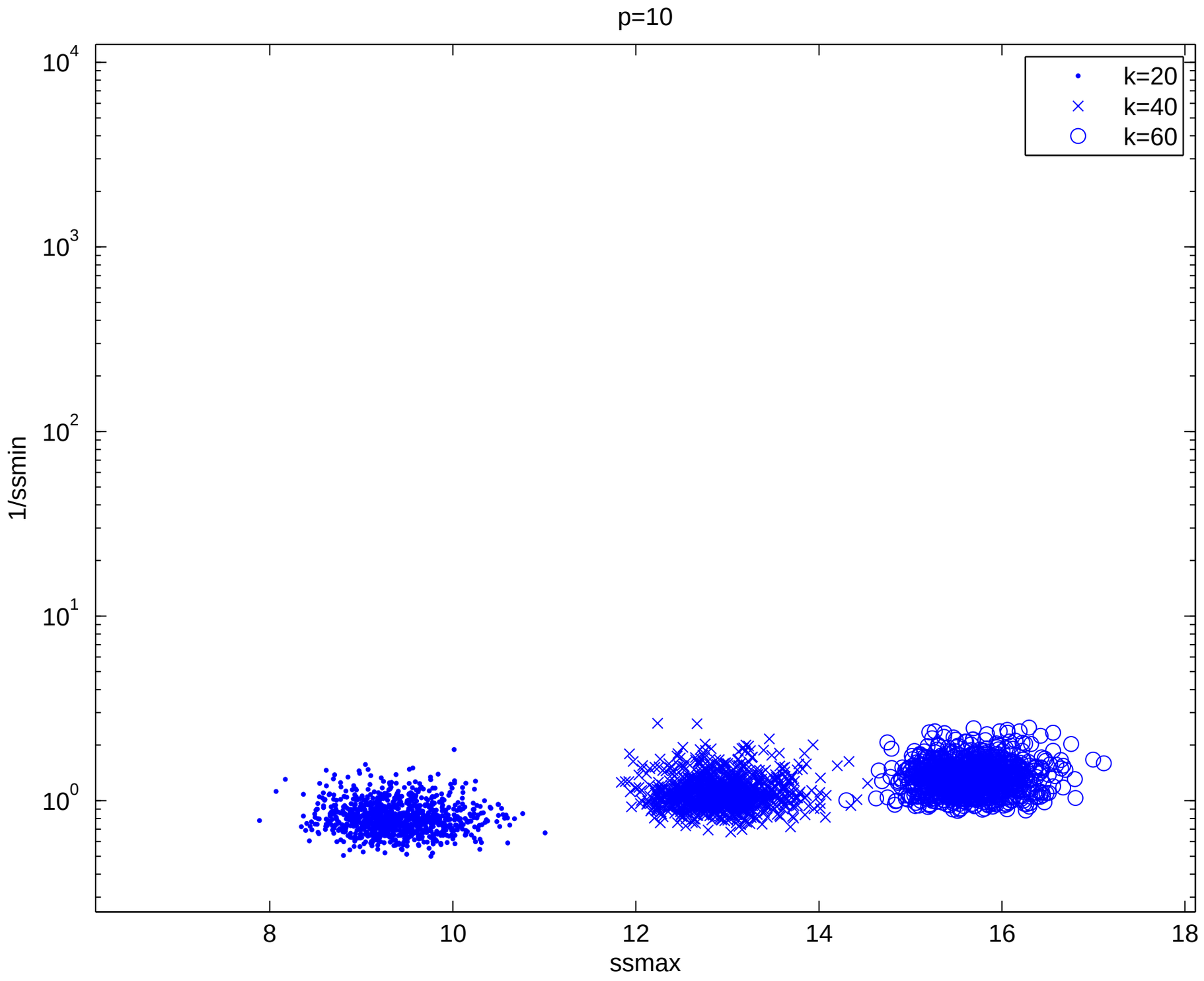
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

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



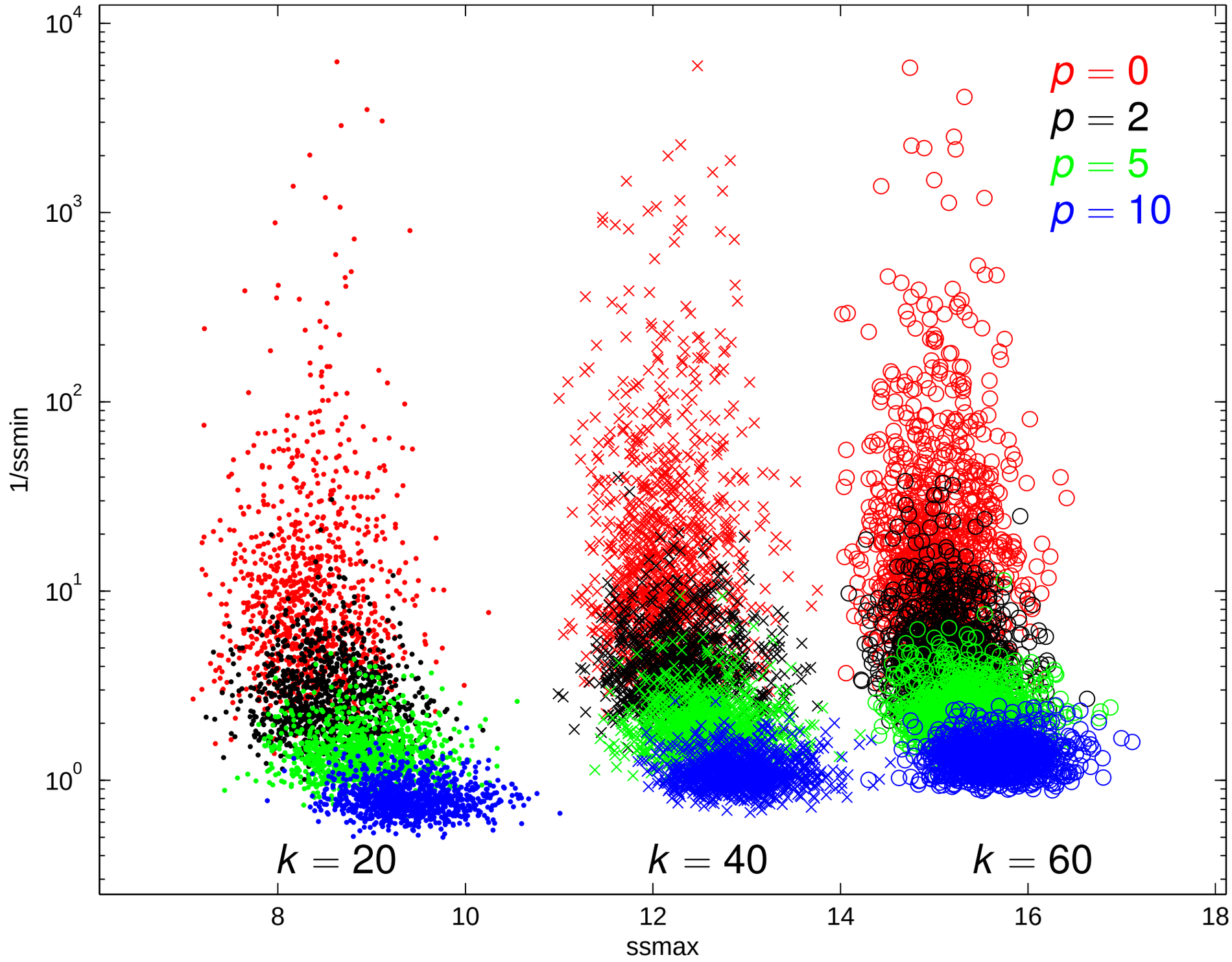
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

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 "r_j^2" denote a generic χ_j^2 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 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: $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|^2$, where $\mathbf{R}_1$ and $\mathbf{R}_2$ are Gaussian and $\mathbf{R}_1$ 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

$$\mathbb{E}\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| = \mathbb{E}\big(\|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|^2\big)^{1/2} \leq \|\mathbf{D}_2\| + \mathbb{E}\|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|.$$

Condition on $\mathbf{R}_1$ and use Proposition 1:

$$\begin{aligned}\mathbb{E}\|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\| &\leq \mathbb{E}\big[\|\mathbf{D}_2\|\,\|\mathbf{R}_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\mathbf{R}_1^\dagger\|\big] \\ &\leq \{\text{Hölder}\} \leq \|\mathbf{D}_2\|\left(\mathbb{E}\|\mathbf{R}_1^\dagger\|_{\mathrm{F}}^2\right)^{1/2} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\mathbb{E}\|\mathbf{R}_1^\dagger\|.\end{aligned}$$

Proposition 2 now provides bounds for $\mathbb{E}\|\mathbf{R}_1^\dagger\|_{\mathrm{F}}^2$ and $\mathbb{E}\|\mathbf{R}_1^\dagger\|$ and we get

$$\mathbb{E}\|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\| \leq \sqrt{\frac{k}{p-1}}\|\mathbf{D}_2\| + \frac{e\sqrt{k+p}}{p}\|\mathbf{D}_2\|_{\mathrm{F}} = \sqrt{\frac{k}{p-1}}\sigma_{k+1} + \frac{e\sqrt{k+p}}{p}\left(\sum_{j>k}\sigma_j^2\right)^{1/2}.$$

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: $\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|^2 \leq \|\mathbf{D}_2\|^2 + \|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|^2$, where $\mathbf{R}_1$ and $\mathbf{R}_2$ are Gaussian and $\mathbf{R}_1$ 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 $E_t = \left\{\|\mathbf{R}_1\| \leq \frac{e\sqrt{k+p}}{p+1}t \quad \text{and} \quad \|\mathbf{R}_1^\dagger\|_{\mathrm{F}} \leq \sqrt{\frac{3k}{p+1}}\,t\right\}$. By Proposition 4: $\mathbb{P}(E_t^{\mathrm{c}}) \leq 2\,t^{-p}$.

Set $h(\mathbf{X}) = \|\mathbf{D}_2\mathbf{X}\mathbf{R}_1^\dagger\|$. A direct calculation shows

$$|h(\mathbf{X}) - h(\mathbf{Y})| \leq \|\mathbf{D}_2\|\,\|\mathbf{R}_1^\dagger\|\,\|\mathbf{X} - \mathbf{y}\|_{\mathrm{F}}.$$

Hold $\mathbf{R}_1$ fixed and take the expectation on $\mathbf{R}_2$. Then Proposition 1 applies and so

$$\mathbb{E}\big[h(\mathbf{R}_2) \;\big|\; \mathbf{R}_1\big] \leq \|\mathbf{D}_2\|\,\|\mathbf{R}_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\mathbf{R}_1^\dagger\|.$$

Now use Proposition 3 (concentration of measure)

$$\mathbb{P}\Big[\underbrace{\|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\|}_{=h(\mathbf{R}_2)} > \underbrace{\|\mathbf{D}_2\|\,\|\mathbf{R}_1^\dagger\|_{\mathrm{F}} + \|\mathbf{D}_2\|_{\mathrm{F}}\,\|\mathbf{R}_1^\dagger\|}_{=\mathbb{E}[h(\mathbf{R}_2)]} + \underbrace{\|\mathbf{D}_2\|\,\|\mathbf{R}_1^\dagger\|}_{=L}\,u \;\Big|\; E_t\Big] < e^{-u^2/2}.$$

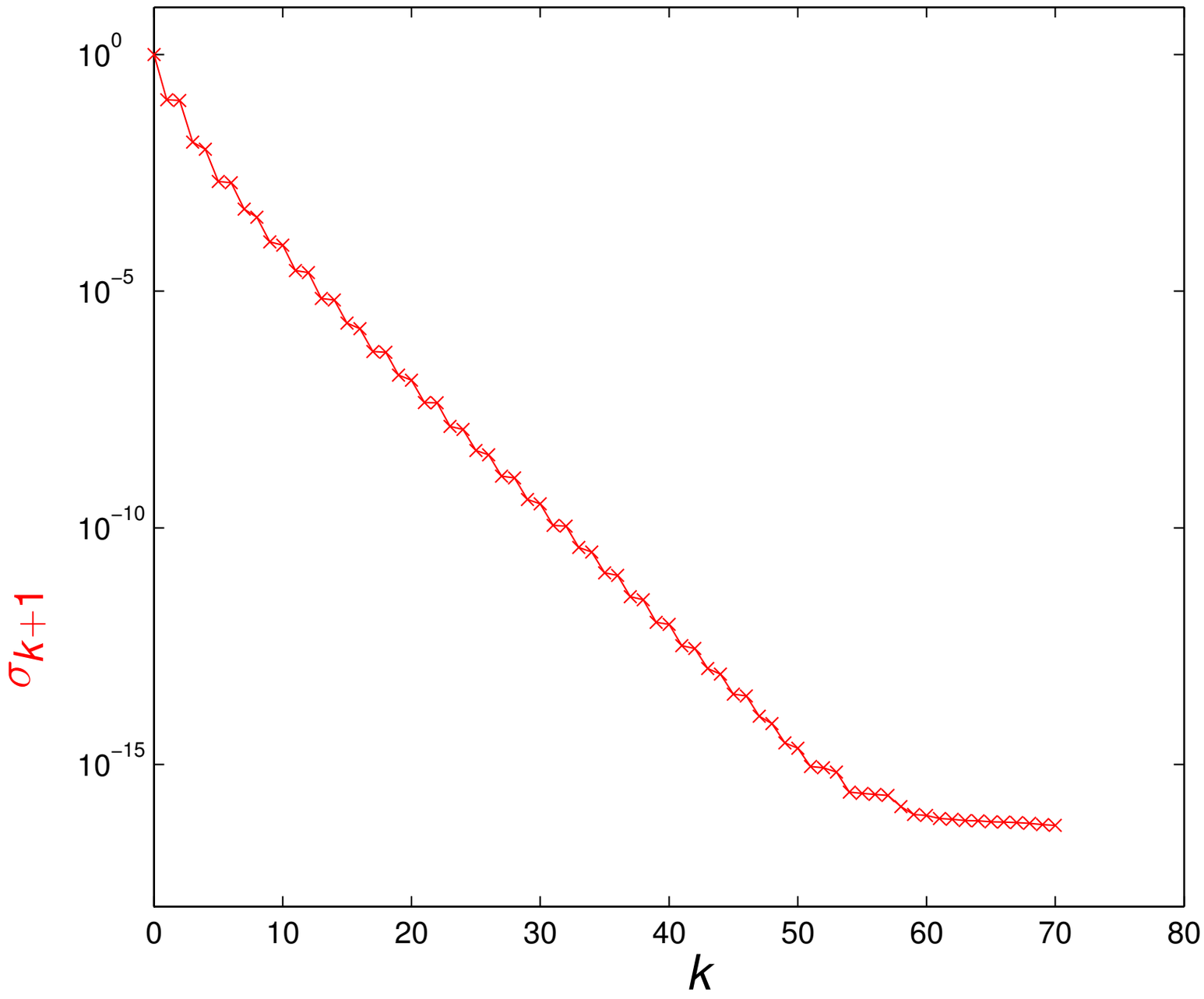
When E_t holds true, we have bounds on the "badness" of $\mathbf{R}_1^\dagger$:

$$\mathbb{P}\left[\|\mathbf{D}_2\mathbf{R}_2\mathbf{R}_1^\dagger\| > \|\mathbf{D}_2\|\sqrt{\frac{3k}{p+1}}t + \|\mathbf{D}_2\|_{\mathrm{F}}\,\frac{e\sqrt{k+p}}{p+1}t + \|\mathbf{D}_2\|\,\frac{e\sqrt{k+p}}{p+1}\,ut \;\middle|\; E_t\right] < e^{-u^2/2}.$$

The theorem is obtained by using $\mathbb{P}(E_t^{\mathrm{c}}) \leq 2\,t^{-p}$ to remove the conditioning of E_t.

Example 1:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



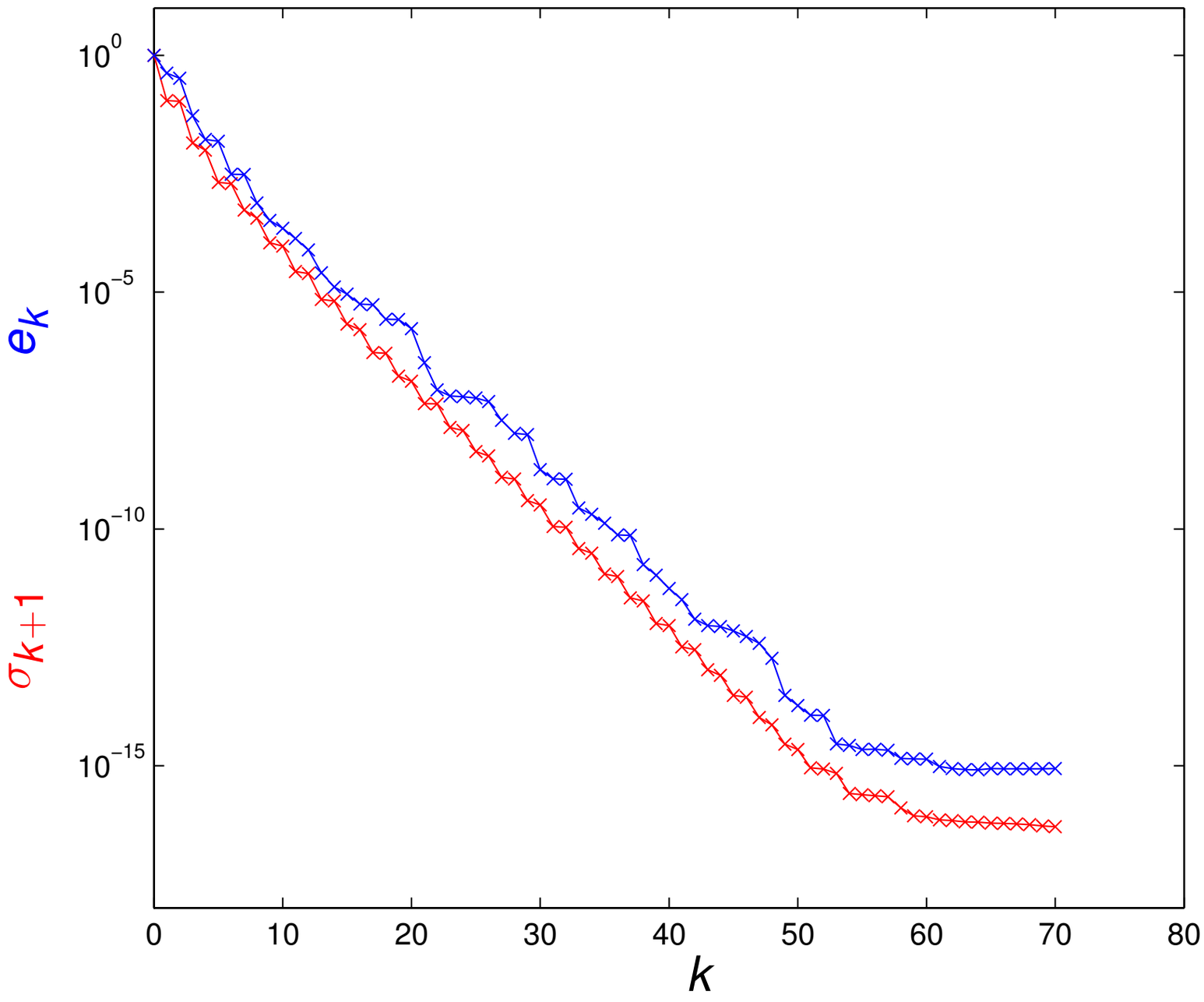
The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 1:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

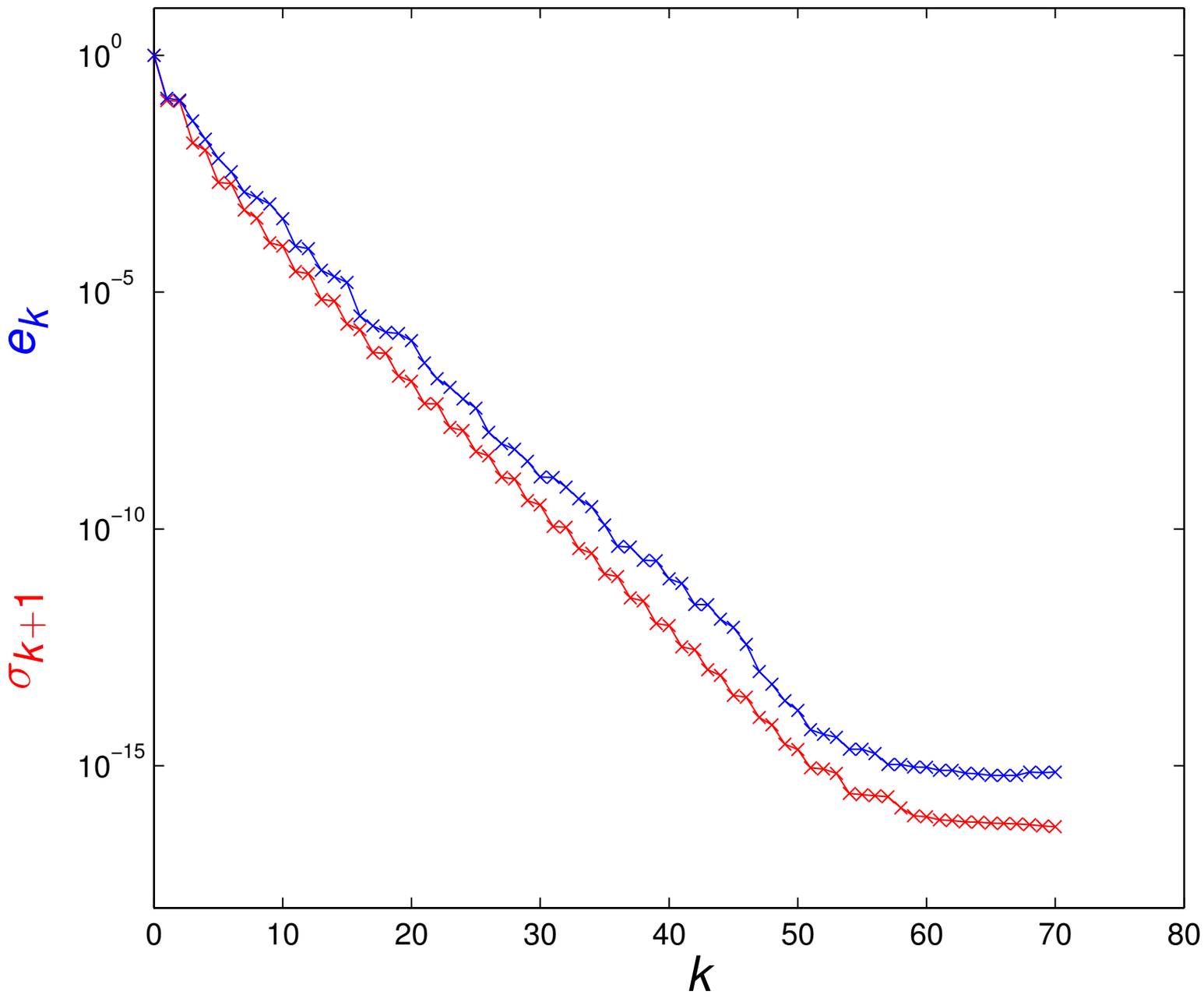
The blue line indicates the actual errors e_k incurred by one instantiation of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 1:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

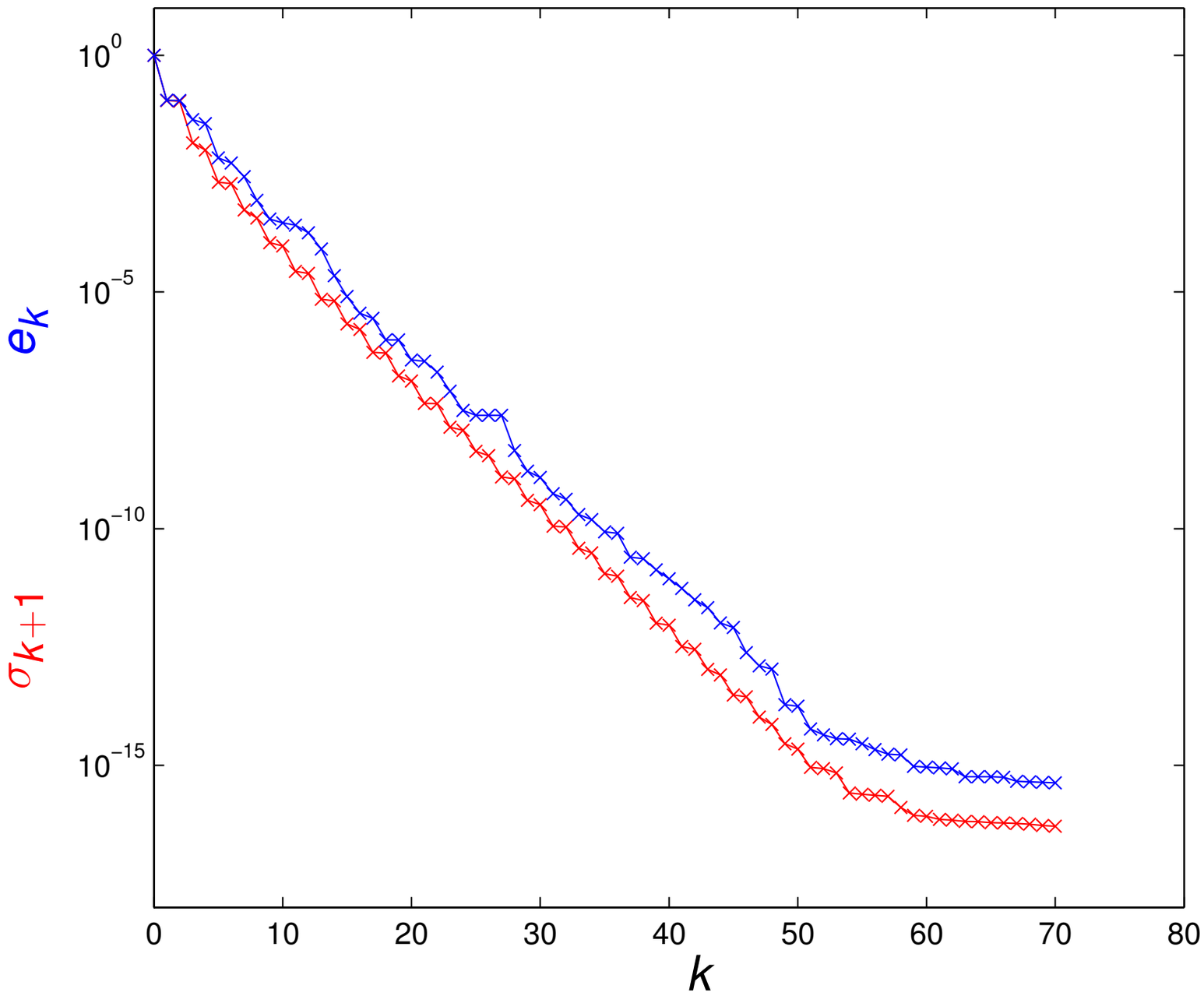
The blue line indicates the actual errors e_k incurred by a different instantiation of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 1:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

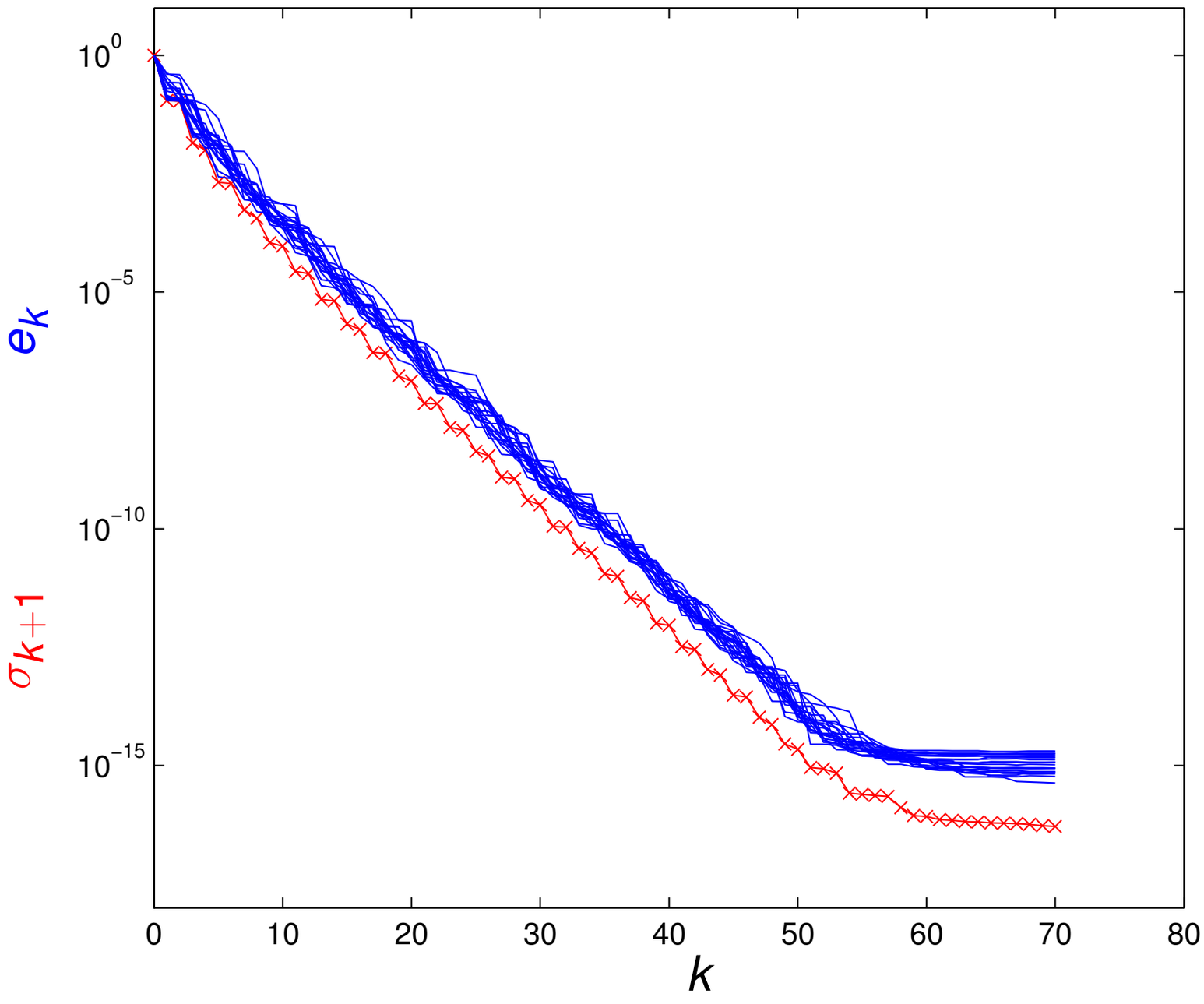
The blue line indicates the actual errors e_k incurred by a different instantiation of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 1:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

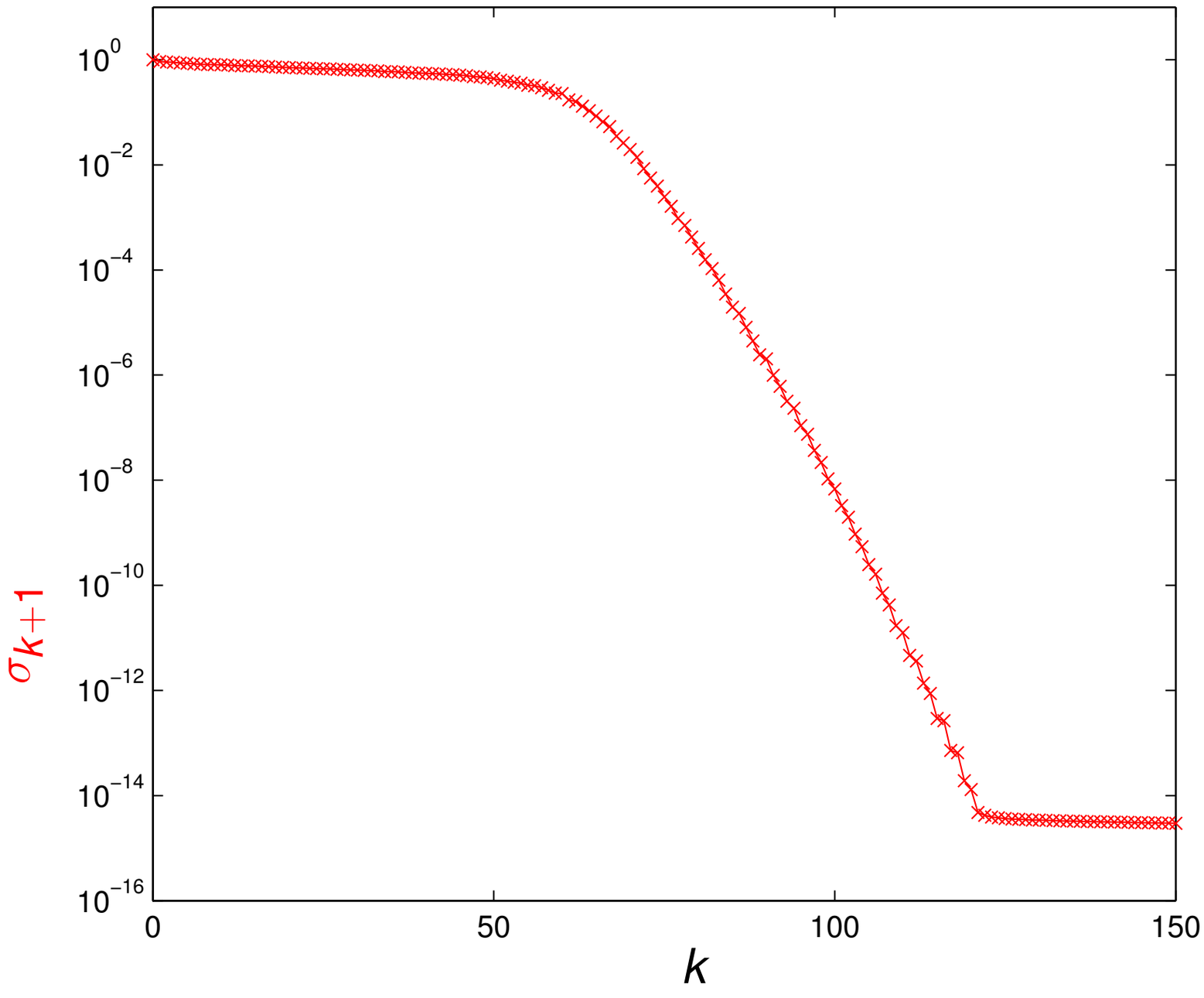
The blue lines indicate the actual errors e_k incurred by 20 instantiations of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 2:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



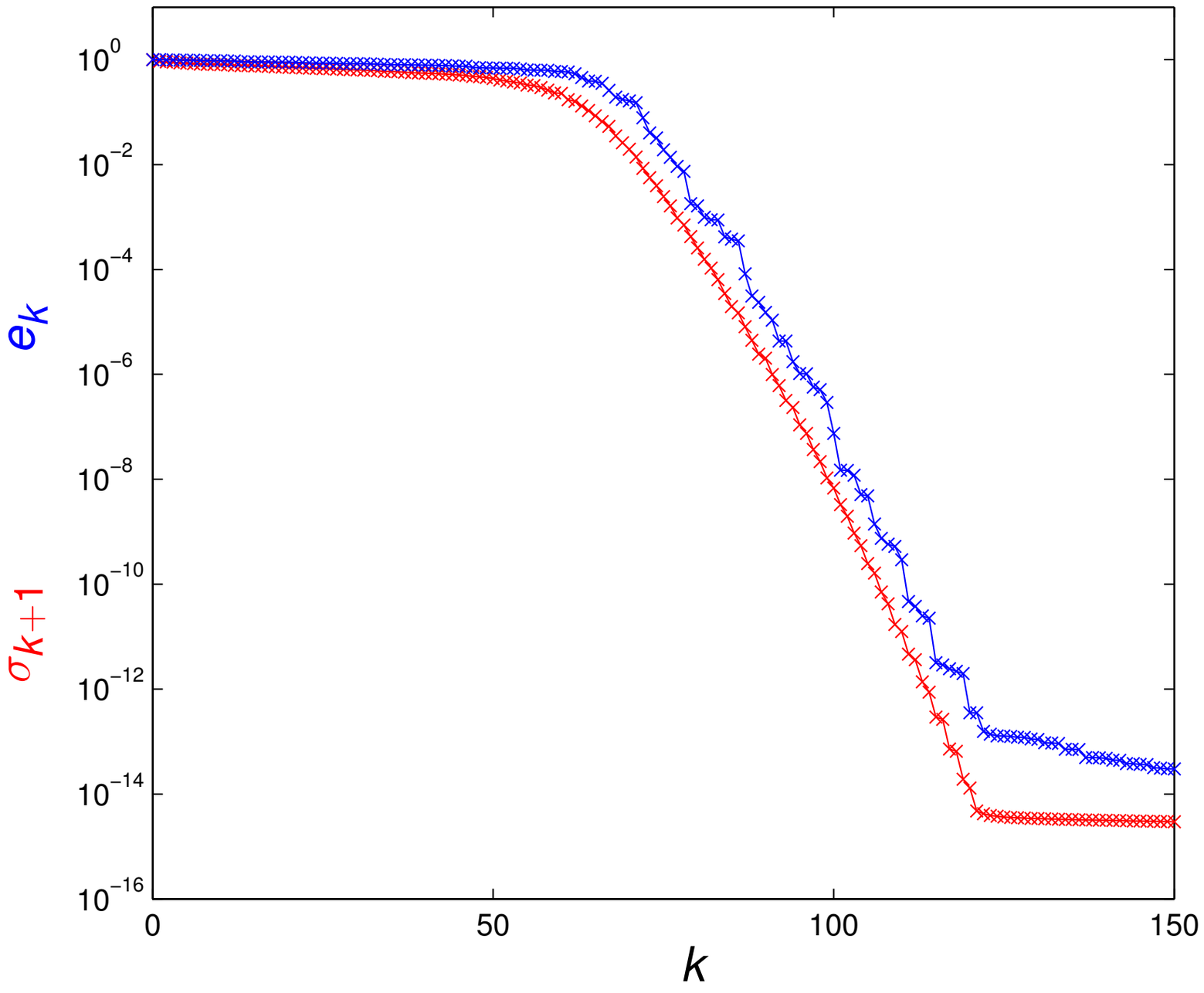
The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 2:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

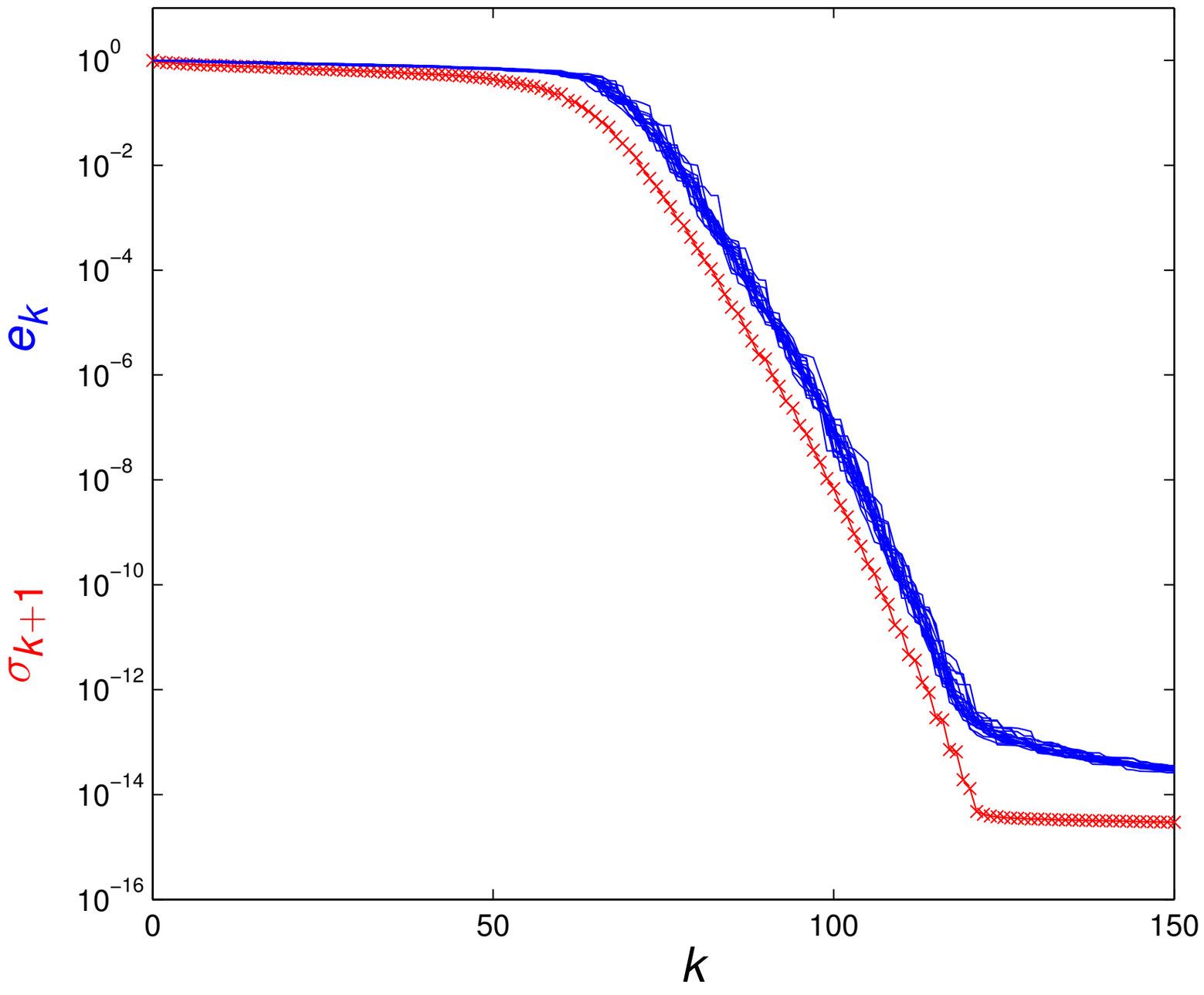
The blue line indicates the actual errors e_k incurred by one instantiation of the proposed method.

A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

Example 2:

We consider a 1 000 $\times$ 1 000 matrix **A** whose singular values are shown below:



The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

The blue lines indicate the actual errors e_k incurred by 20 instantiations of the proposed method.

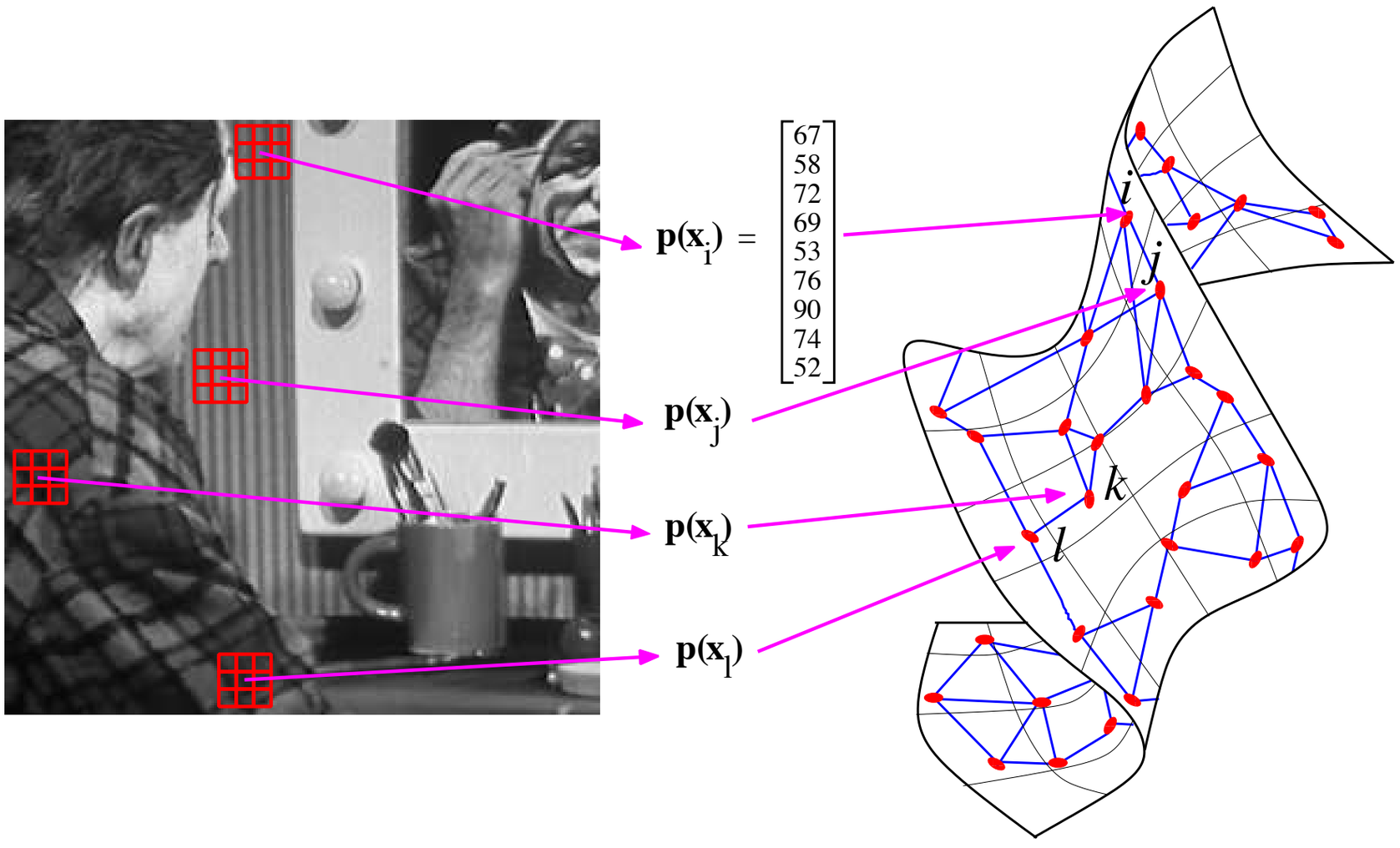
A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

Curiously, the nature of **A** is in a strong sense irrelevant: the error distribution depends only on $\{\sigma_j\}_{j=1}^{\min(m,n)}$.

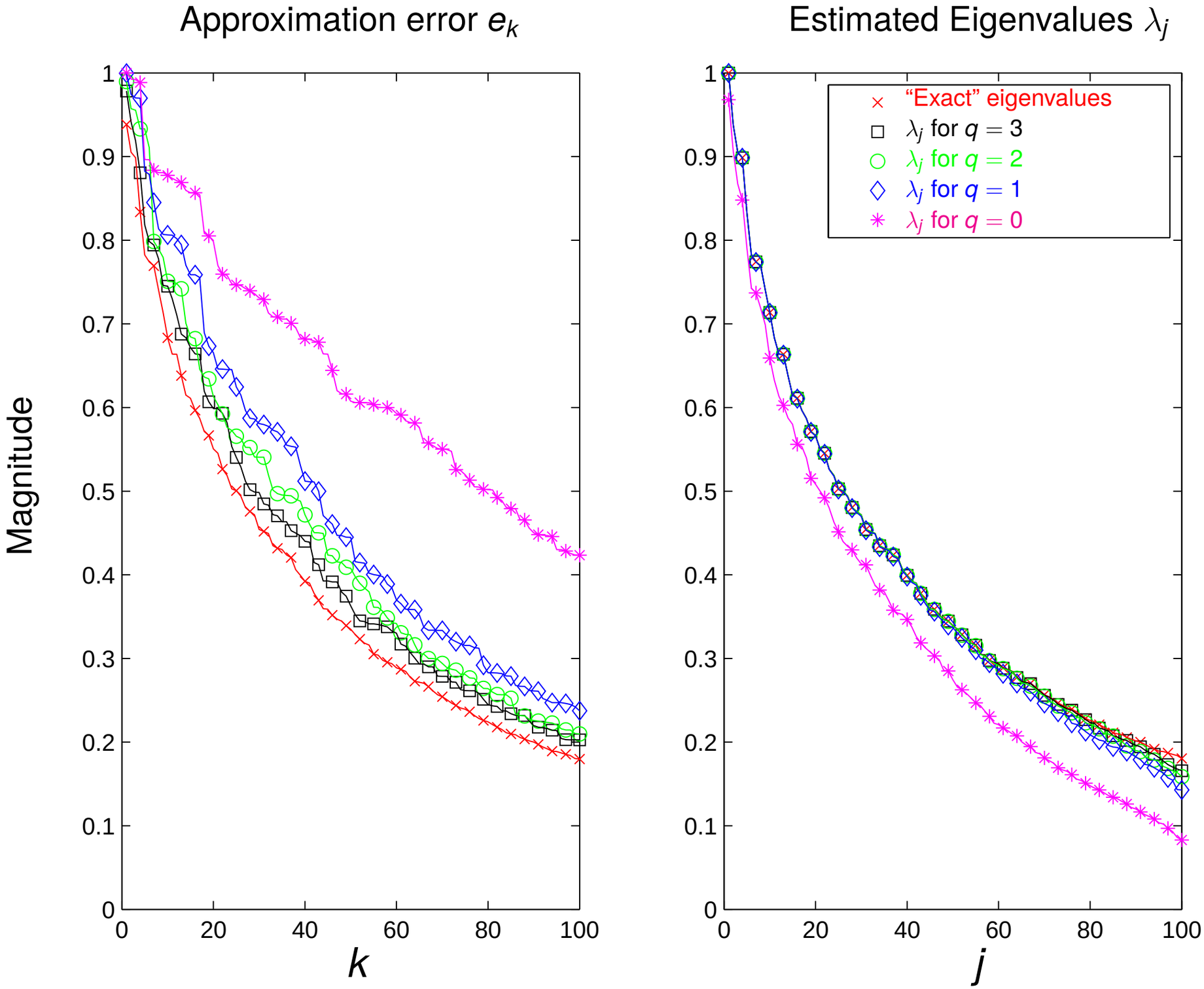
Example 3:

The matrix **A** being analyzed is a 9025 $\times$ 9025 matrix arising in a diffusion geometry approach to image processing.

To be precise, **A** is a graph Laplacian on the manifold of 3 $\times$ 3 patches.



Joint work with François Meyer of the University of Colorado at Boulder.



The pink lines illustrates the performance of the basic random sampling scheme.

The errors are huge, and the estimated eigenvalues are much too small.

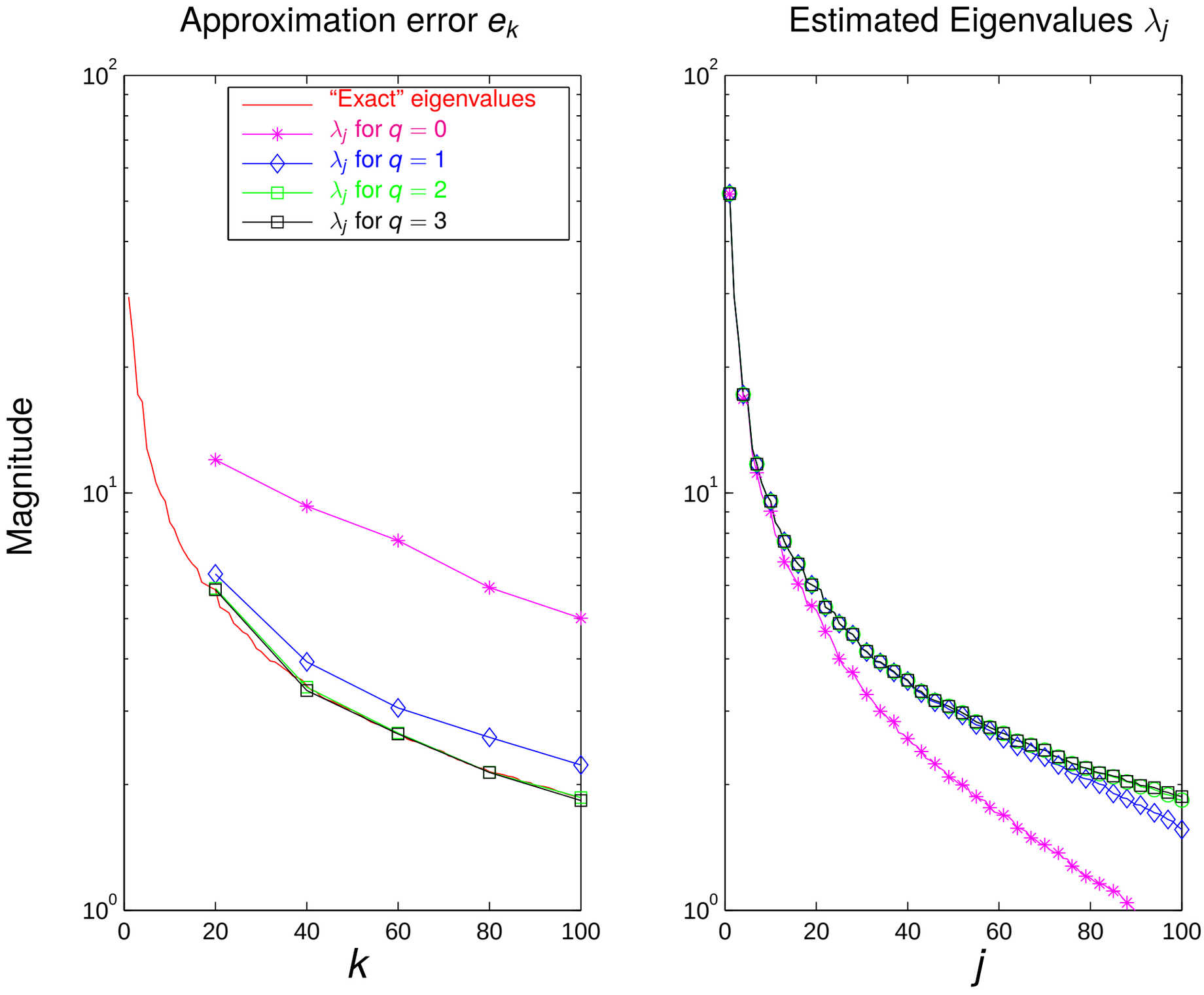
Example 4: “Eigenfaces”

We next process a data base containing $m = 7\,254$ pictures of faces

Each image consists of $n = 384 \times 256 = 98\,304$ gray scale pixels.

We center and scale the pixels in each image, and let the resulting values form a column of a $98\,304 \times 7\,254$ data matrix $\mathbf{A}$.

The left singular vectors of $\mathbf{A}$ are the so called *eigenfaces* of the data base.



The pink lines illustrates the performance of the basic random sampling scheme. Again, the errors are huge, and the estimated eigenvalues are much too small.

Power method for improving accuracy:

The error depends on how quickly the singular values decay. Recall that

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

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 singular values

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

Much faster decay — so let us use the sample matrix

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

instead of

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

References: Paper by Rokhlin, Szlam, Tygert (2008). Suggestions by Ming Gu. Also similar to “block power method,” “block Lanczos,” “subspace iteration.”

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 $n \times \ell$ random matrix **R**.	(4) Form the small matrix $\mathbf{B} = \mathbf{Q}^* \mathbf{A}$.
(2) Form the $m \times \ell$ sample matrix $\mathbf{Y} = (\mathbf{A}\,\mathbf{A}^*)^q\,\mathbf{AR}$.	(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:

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

  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 (1) 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.

A numerically stable version of the "power method":

Input: An $m \times n$ matrix $\mathbf{A}$, a target rank ℓ, and a small integer q.

Output: Rank-ℓ factors $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ in an approximate SVD $\mathbf{A} \approx \mathbf{UDV}^*$.

Draw an $n \times \ell$ Gaussian random matrix $\mathbf{R}$.
Set $\mathbf{Q} = \mathtt{orth}(\mathbf{AR})$
for $i = 1, 2, \ldots, q$
  $\mathbf{W} = \mathtt{orth}(\mathbf{A}^*\mathbf{Q})$
  $\mathbf{Q} = \mathtt{orth}(\mathbf{AW})$
end for
$\mathbf{B} = \mathbf{Q}^*\mathbf{A}$
$[\hat{\mathbf{U}}, \mathbf{D}, \mathbf{V}] = \mathtt{svd}(\mathbf{B})$
$\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$.

Note: Algebraically, the method with orthogonalizations is identical to the "original" method where $\mathbf{Q} = \mathtt{orth}((\mathbf{AA}^*)^q\mathbf{AR})$.

Note: This is a classic subspace iteration.

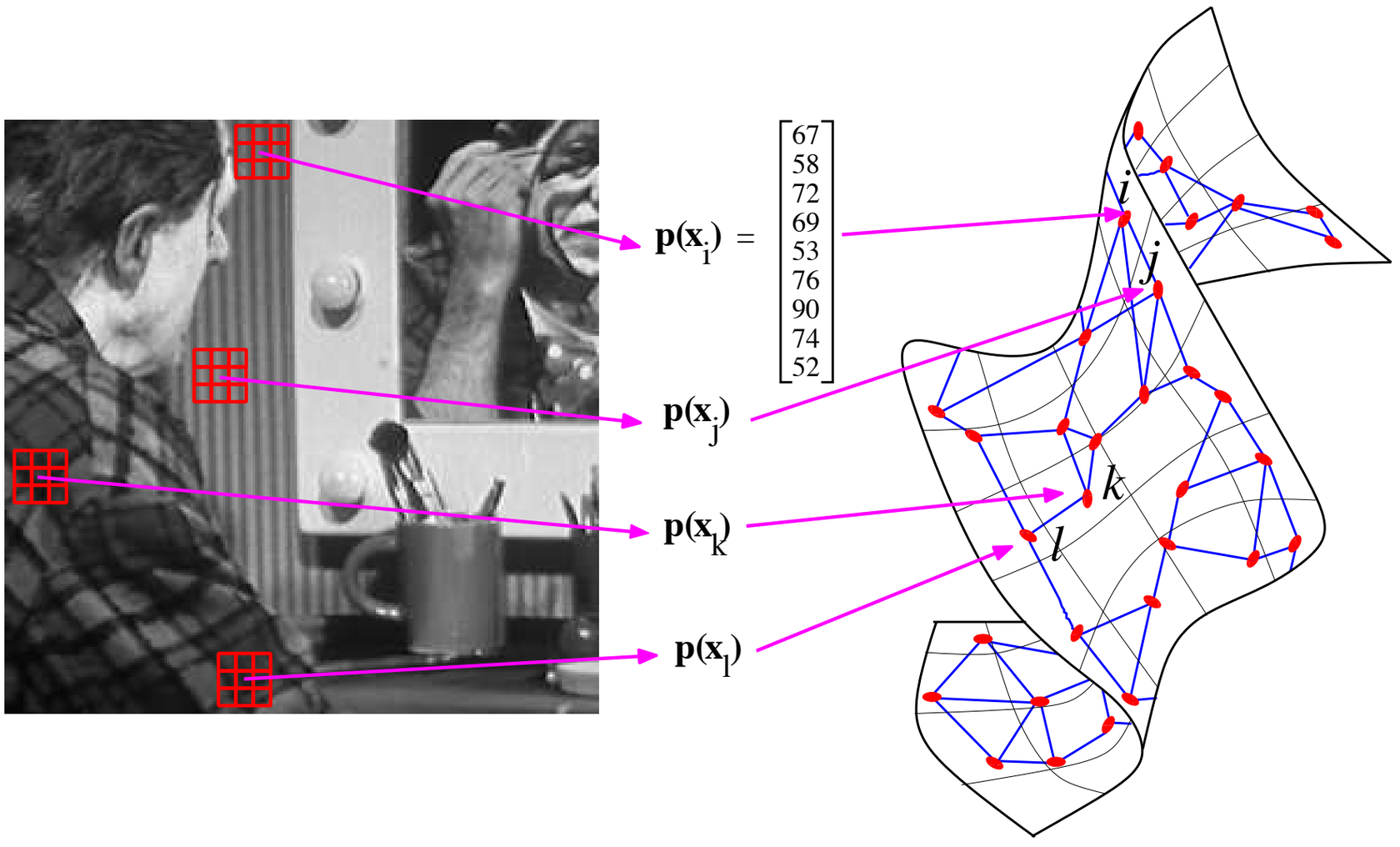
The novelty is the error analysis, and the finding that using a very small q is often fine.

(In fact, our analysis allows q to be zero...)

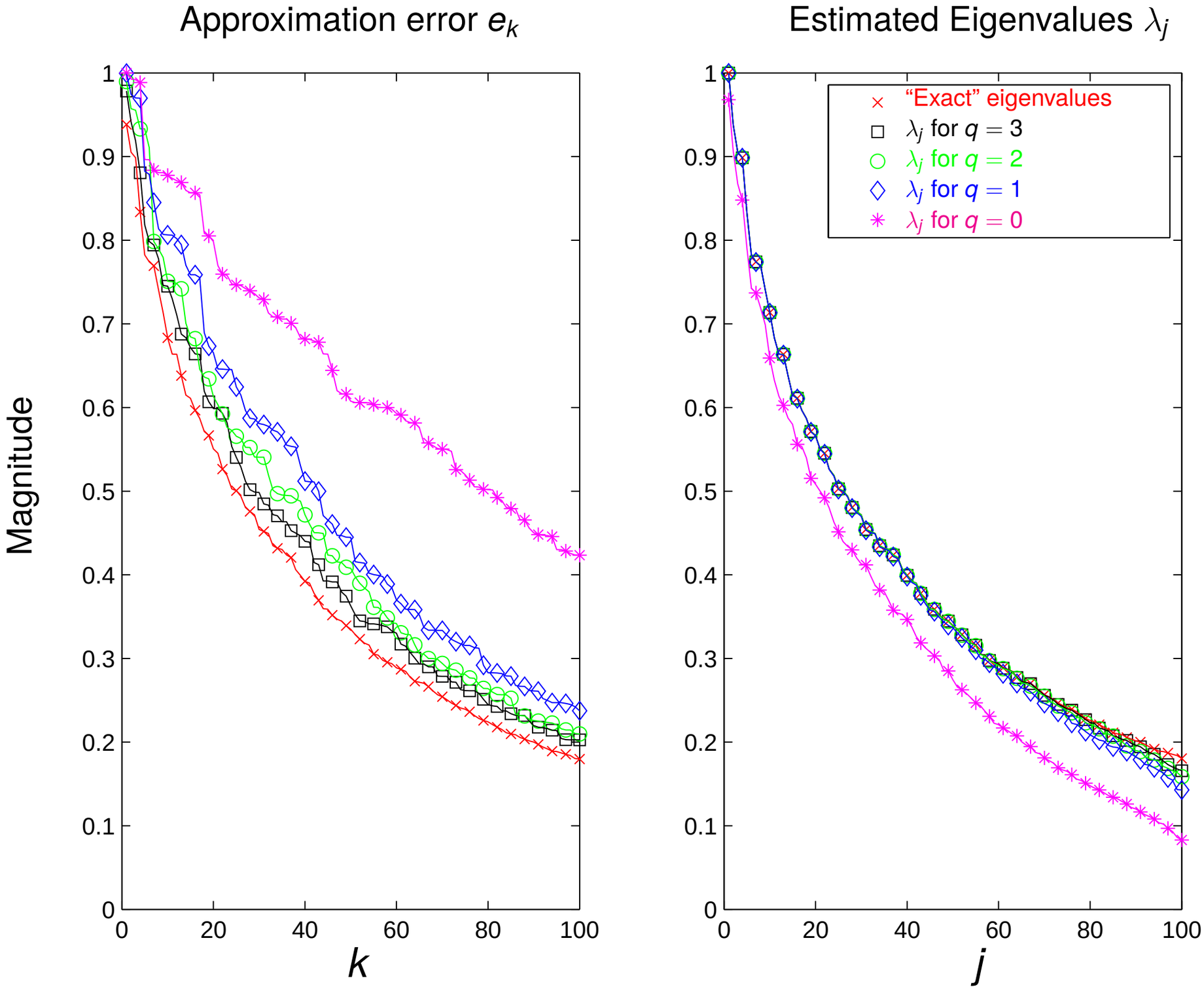
Example 3 (revisited):

The matrix **A** being analyzed is a 9025 $\times$ 9025 matrix arising in a diffusion geometry approach to image processing.

To be precise, **A** is a graph Laplacian on the manifold of 3 $\times$ 3 patches.



Joint work with François Meyer of the University of Colorado at Boulder.



The pink lines illustrates the performance of the basic random sampling scheme.

The errors for $q = 0$ are huge, and the estimated eigenvalues are much too small.

But: The situation improves very rapidly as q is cranked up!

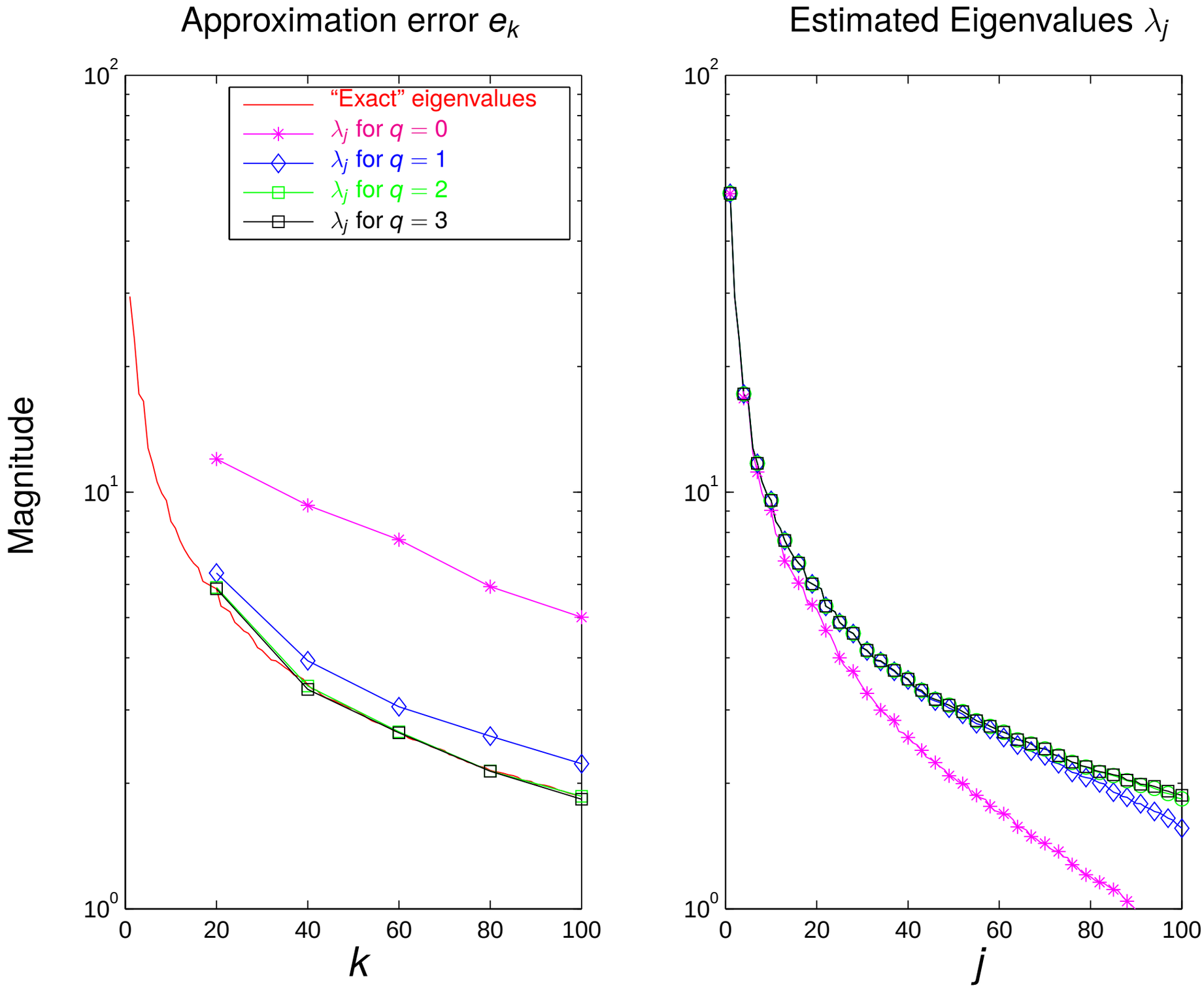
Example 4 (revisited): “Eigenfaces”

We next process a data base containing $m = 7\,254$ pictures of faces

Each image consists of $n = 384 \times 256 = 98\,304$ gray scale pixels.

We center and scale the pixels in each image, and let the resulting values form a column of a $98\,304 \times 7\,254$ data matrix $\mathbf{A}$.

The left singular vectors of $\mathbf{A}$ are the so called *eigenfaces* of the data base.



The pink lines illustrates the performance of the basic random sampling scheme.

Again, the errors are huge, and the estimated eigenvalues are much too small.

But: The situation improves very rapidly as q is cranked up!

Current work:

The material presented so far is fairly well established by now. (Recycled slides . . .)

Next, let us briefly describe current research in this area.

Current work: Randomized approximation of "rank-structured" matrices

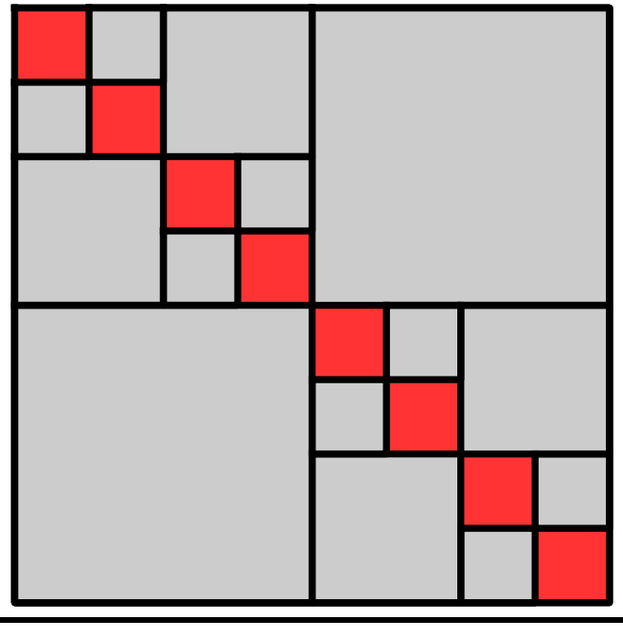
Many matrices in applications have *off-diagonal blocks* that are of low rank:

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

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

- *Accelerated direct solvers for elliptic PDEs.*
- *$O(N)$ complexity in many situations.*

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



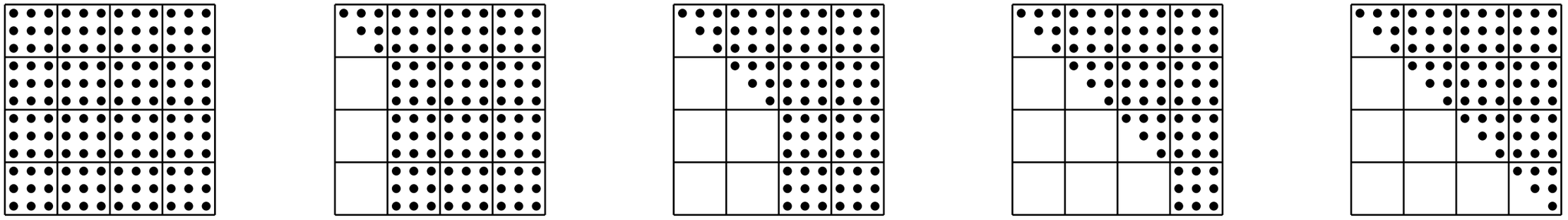
Current work: Accelerate FULL factorizations of matrices

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.

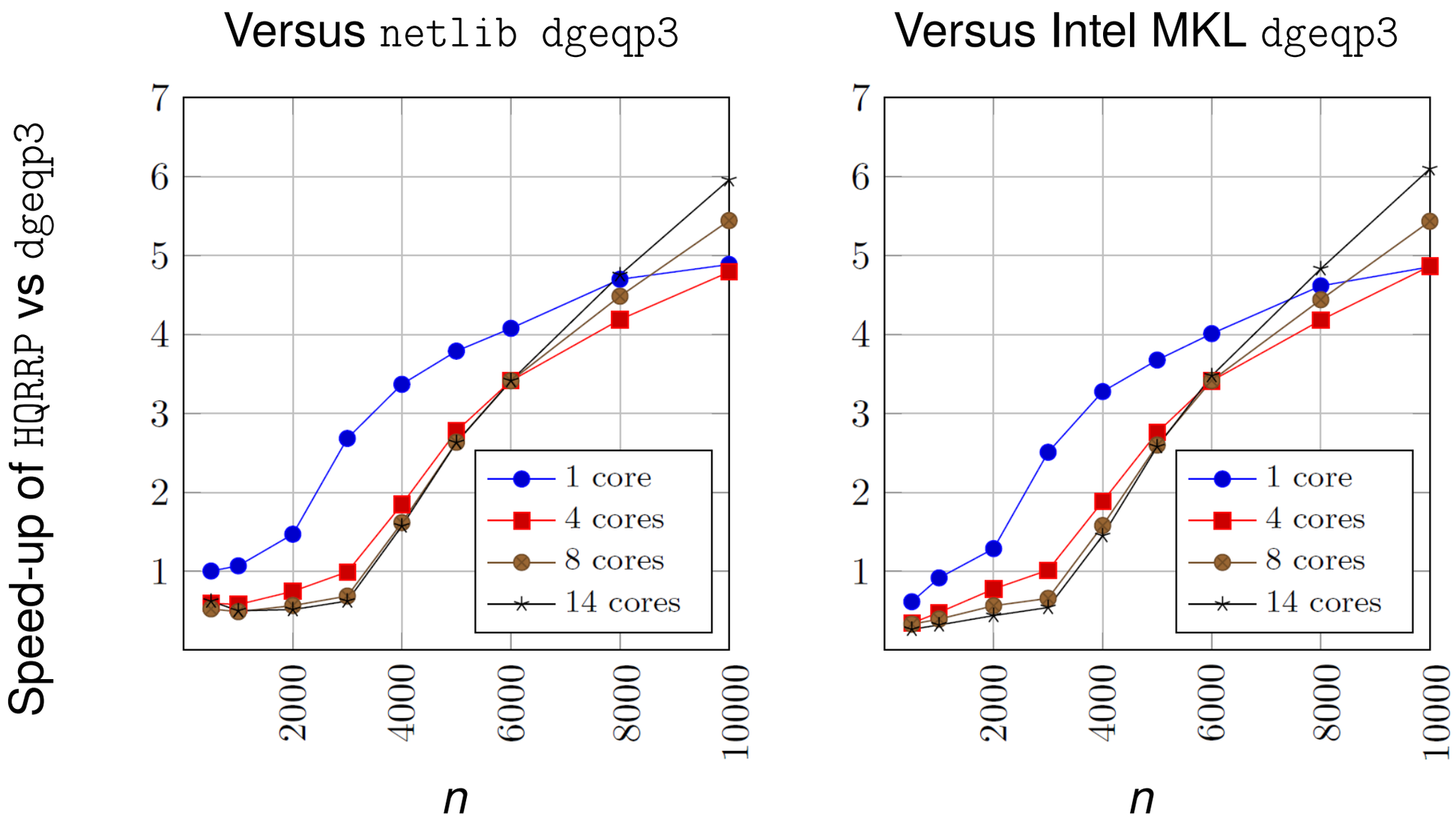
Each $\mathbf{Q}_j$ is a product of Householder reflectors.

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.

Current work: Accelerate FULL 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/`

Current work: Accelerate FULL factorizations of matrices

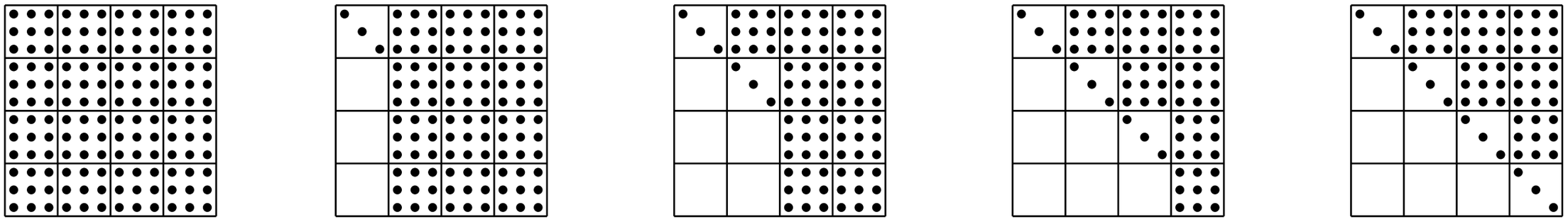
Given a dense $n \times n$ matrix $\mathbf{A}$, compute a factorization

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

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:



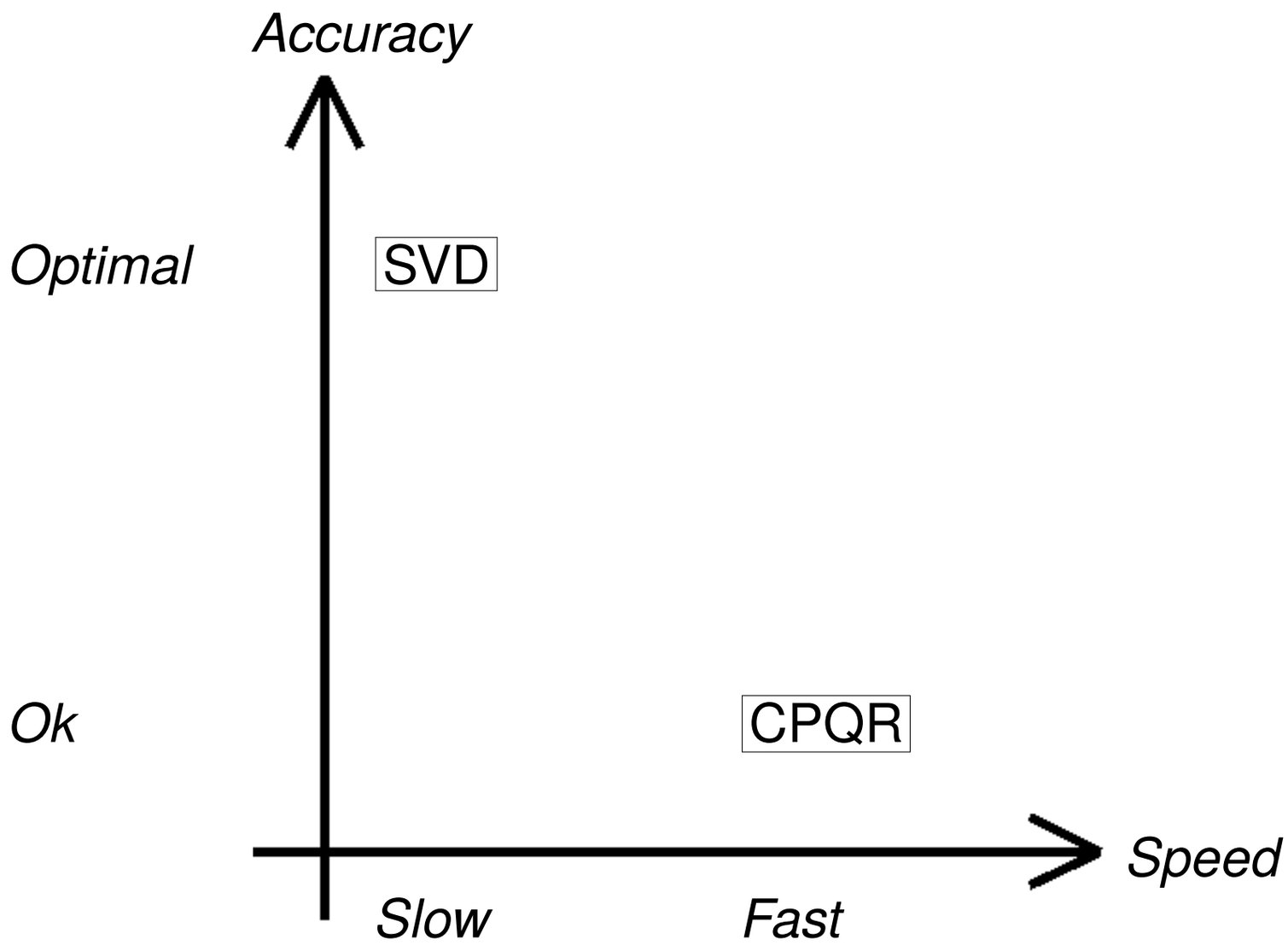
$$\mathbf{A}_0 = \mathbf{A} \qquad \mathbf{A}_1 = \mathbf{U}_1^* \mathbf{A}_0 \mathbf{V}_1 \qquad \mathbf{A}_2 = \mathbf{U}_2^* \mathbf{A}_1 \mathbf{V}_2 \qquad \mathbf{A}_3 = \mathbf{U}_3^* \mathbf{A}_2 \mathbf{V}_3 \qquad \mathbf{A}_4 = \mathbf{U}_4^* \mathbf{A}_3 \mathbf{V}_4$$

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.

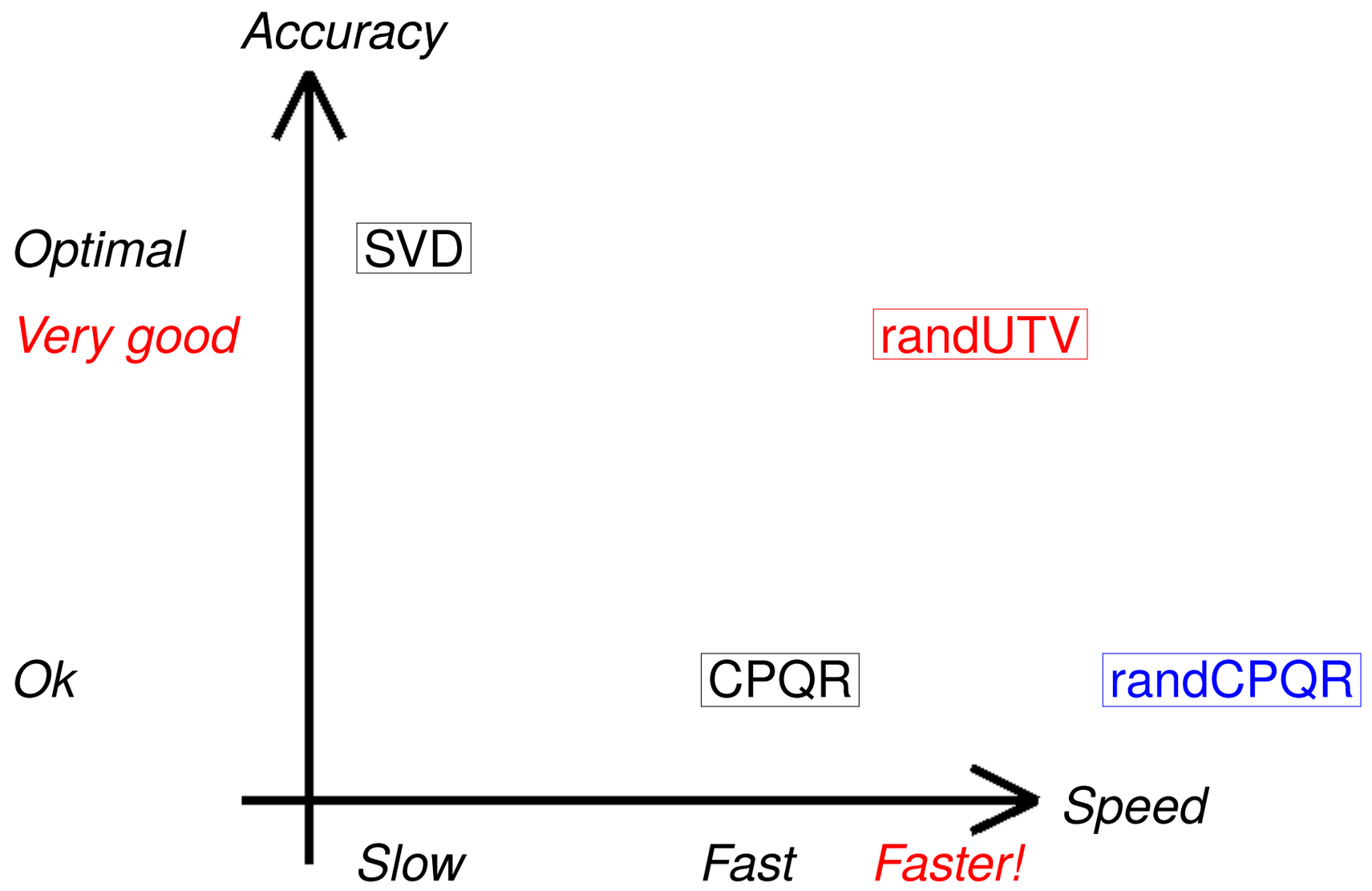
Current work: Accelerate FULL factorizations of matrices

For the task of computing low-rank approximations to matrices, the classical choice is between SVD and column pivoted QR (CPQR). SVD is slow, and CPQR is inaccurate:



Current work: Accelerate FULL factorizations of matrices

For the task of computing low-rank approximations to matrices, the classical choice is between SVD and column pivoted QR (CPQR). SVD is slow, and CPQR is inaccurate:



The randomized algorithm `randUTV` combines the best properties of both factorizations. Additionally, `randUTV` parallelizes better, and allows the computation of partial factorizations (like CPQR, but unlike SVD).

Current work: Randomized pre-conditioners, nearest neighbor search, clustering, . . .

Question: Is it possible to build algorithms that combine the powerful dimension reduction capability of randomized projections with the accuracy and robustness of classical deterministic methods?

Putative answer: Yes — use a two-stage approach:

(A) *Randomized pre-conditioner:*

In a pre-computation, random projections are used to create low-dimensional sketches of the high-dimensional data. These sketches are somewhat distorted, but approximately preserve key properties to very high probability.

(B) *Deterministic post-processing:*

Once a sketch of the data has been constructed in Stage A, classical deterministic techniques are used to compute desired quantities to very high accuracy, *starting directly from the original high-dimensional data*.

It is often advantageous to add a final step of *à posteriori error estimation.*
This can typically be done very cheaply using randomized sampling.

Current work: Randomized pre-conditioners, nearest neighbor search, clustering, . . .

Example 1 of two-stage approach: Randomized SVD

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

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

where **U** and **V** are orthonormal, and **D** is diagonal.

(A) *Randomized pre-conditioner:*

Use randomized projection methods to form an approximate basis for the range of the matrix.

(B) *Deterministic post-processing:*

Restrict the matrix to the subspace determined in Stage A, and perform expensive but accurate computations on the resulting smaller matrix.

Observe that distortions in the randomized projections are fine, since all we need is a subspace the captures "most" of the range. Pollution from unwanted singular modes is harmless, as long as we capture the dominant ones. The risk of missing the dominant ones is for practical purposes zero.

Current work: Randomized pre-conditioners, nearest neighbor search, clustering, ...

Example 1 of two-stage approach: Randomized SVD

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

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

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, and $\mathbf{D}$ is diagonal.

Fix an over-sampling parameter p. Say $p = 10$.

(A) *Randomized pre-conditioner:*

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

(B) *Deterministic post-processing:*

- B.1 Form the $(k+p) \times n$ matrix $\mathbf{B} = \mathbf{Q}^*\,\mathbf{A}$. `B = Q' * A`
- B.2 Form SVD of the matrix $\mathbf{B}$: $\mathbf{B} = \hat{\mathbf{U}}\,\mathbf{D}\,\mathbf{V}^*$. `[Uhat, Sigma, V] = svd(B,0)`
- B.3 Form the matrix $\mathbf{U} = \mathbf{Q}\,\hat{\mathbf{U}}$. `U = Q * Uhat`

(Truncate the last p terms in step B.2 to attain a factorization of precise rank k.)

Current work: Randomized pre-conditioners, nearest neighbor search, clustering, . . .

Example 2 of two-stage approach: Nearest neighbor search in $\mathbb{R}^D$ (Jones, Osipov, Rokhlin)

Objective: Suppose you are given n points $\{\boldsymbol{x}_j\}_{j=1}^n$ in $\mathbb{R}^D$. The coordinate matrix is

$$\mathbf{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots \boldsymbol{x}_n] \in \mathbb{R}^{D\times n}.$$

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

If D is "small" (say $D \leq 10$ 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 D 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.

Current work: Randomized pre-conditioners, nearest neighbor search, clustering, . . .

Example 2 of two-stage approach: Nearest neighbor search in $\mathbb{R}^D$ (Jones, Osipov, Rokhlin)

Objective: Suppose you are given n points $\{\boldsymbol{x}_j\}_{j=1}^n$ in $\mathbb{R}^D$. The coordinate matrix is

$$\mathbf{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots \boldsymbol{x}_n] \in \mathbb{R}^{D\times n}.$$

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

(A) *Randomized probing of data:*

Use a Johnson-Lindenstrauss random projection to map the n-particle problem in $\mathbb{R}^D$ (where D is large) to an n-particle problem in $\mathbb{R}^d$ where $d \sim \log n$. Run a deterministic nearest-neighbor search in $\mathbb{R}^d$ and store a list of the ℓ nearest neighbors for each particle (for simplicity, one can set $\ell = k$). Then repeat the process several times. If for a given particle a previously undetected neighbor is discovered, then simply add it to a list of potential neighbors.

(B) *Deterministic post-processing:*

The randomized probing will result in a list of putative neighbors that typically contains more than k elements. But it is now easy to compute the pairwise distances in the original space $\mathbb{R}^D$ to judge which candidates in the list are the k nearest neighbors.

Current work on randomized projections:

1. *Accelerate full factorizations of matrices.*

   New randomized column pivoted QR algorithm is much faster than LAPACK.

   New "UTV" factorization method is almost as accurate as SVD and much faster.

2. *Randomized algorithms for structured matrices.*

   Use randomization to accelerate key numerical solvers for PDEs, for simulating Gaussian processes, etc.

3. *Use randomized projections to accelerate non-linear algebraic tasks.*

   Faster nearest neighbor search, faster clustering algorithms, etc. The idea is to use randomized projections for *sketching* to develop a rough map of a large data set. Then use high-accuracy deterministic methods for the actual computation.

4. *[High risk/high reward] Accelerate linear solvers for "general" systems* $\mathbf{Ax} = \mathbf{b}$*.*

   The goal is methods with complexity $O(N^\gamma)$ for $\gamma < 3$. Crucially, we seek methods that retain stability, and have high practical efficiency for realistic problem sizes. (Cf. Strassen — $O(N^{2.81})$, Coppersmith-Winograd $O(N^{2.38})$, etc.)

Current work on randomized projections:

1. *Accelerate full factorizations of matrices.*

   New randomized column pivoted QR algorithm is much faster than LAPACK.

   New "UTV" factorization method is almost as accurate as SVD and much faster.

2. *Randomized algorithms for structured matrices.*

   Use randomization to accelerate key numerical solvers for PDEs, for simulating Gaussian processes, etc.

3. *Use randomized projections to accelerate non-linear algebraic tasks.*

   Faster nearest neighbor search, faster clustering algorithms, etc. The idea is to use randomized projections for *sketching* to develop a rough map of a large data set. Then use high-accuracy deterministic methods for the actual computation.

4. *[High risk/high reward] Accelerate linear solvers for "general" systems* $\mathbf{Ax} = \mathbf{b}$*.*

   The goal is methods with complexity $O(N^\gamma)$ for $\gamma < 3$. Crucially, we seek methods that retain stability, and have high practical efficiency for realistic problem sizes. (Cf. Strassen — $O(N^{2.81})$, Coppersmith-Winograd $O(N^{2.38})$, etc.)


Great potential for new discoveries in linear algebra!

Final remarks:

- For large scale SVD/PCA of dense matrices, these algorithms are highly recommended; they compare favorably to existing methods in almost every regard.
- 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.* $\eta = 10^{-10}$ or 10^{-20}).
- This talk 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.
- Randomized methods for computing "FMM"-style (HSS, $\mathcal{H}$-matrix, ...) representations of matrices exist — [M— 2008, 2011, 2015], [Lin, Lu, Ying 2011].
  Leads to accelerated, often $O(N)$, *direct solvers* for elliptic PDEs.
  Applications to scattering, composite materials, engineering design, etc.

Tutorials, summer schools, etc:

- 2009: NIPS tutorial lecture, Vancouver, 2009. Online video available.
- 2014: CBMS summer school at Dartmouth College. 10 lectures on YouTube.
- 2016: Park City Math Institute (IAS): *The Mathematics of Data.*

Software packages:

- Column pivoted QR: `https://github.com/flame/hqrrp` (much faster than LAPACK!)
- Randomized UTV: `https://github.com/flame/randutv`
- RSVDPACK: `https://github.com/sergeyvoronin` (expansions are in progress)
- ID: `http://tygert.com/software.html`

Papers (see also `http://people.maths.ox.ac.uk/martinsson/main_publications.html`):

- P.G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the approximation of matrices". 2007 report YALE-CS-1361; 2011 paper in ACHA.
- N. Halko, P.G. Martinsson, J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Review*, 2011.
- E. Liberty, F. Woolfe, P.G. Martinsson, V. Rokhlin, and M. Tygert, "Randomized algorithms for the low-rank approximation of matrices". *PNAS*, **104**(51), 2007.
- P.G. Martinsson, "A fast randomized algorithm for computing a Hierarchically Semi-Separable representation of a matrix". *SIMAX*, **32**(4), 2011.
- P.G. Martinsson, "Compressing structured matrices via randomized sampling," SISC **38**(4), 2016.
- P.G. Martinsson, G. Quintana-Ortí, N. Heaver, and R. van de Geijn, "Householder QR Factorization With Randomization for Column Pivoting." SISC, **39**(2), pp. C96-C115, 2017.