Randomized methods for accelerating matrix factorization algorithms

Gunnar Martinsson

Mathematical Institute, University of Oxford

Collaborators: Robert van de Geijn, Adrianna Gillman, Nathan Heavner (PhD student), Grigorio Quintana-Ortí, Sergey Voronin (former postdoc).

Slides: Google “Gunnar Martinsson,” then go to “Talks” tab.

Research support by:

Synopsis: The talk will describe techniques for computing a low-rank approximation to a dense matrix through the use of randomized projections.

Environment 1: Given a dense $m \times n$ matrix $\mathbf{A}$ (whose singular values decay), compute an approximate factorization

$$\begin{array}{ccccc} \mathbf{A} & \approx & \mathbf{Q} & & \mathbf{B} \\ m \times n & & m \times k & & k \times n \end{array}$$

where $k \ll \min(min)$. Typically, we are given a tolerance and need to determine k.

Environment 2: Given a dense $m \times n$ matrix $\mathbf{A}$ (with $m \geq n$) compute QR factorization

$$\begin{array}{ccccc} \mathbf{A} & \mathbf{P} & \approx & \mathbf{Q} & \mathbf{R} \\ m \times n & n \times n & & m \times k & k \times n \end{array}$$

for either $k = n$ (full factorization) or k comparable to $\min(m, n)$.

Environment 3: Given a *rank-structured* matrix $\mathbf{A}$ (HSS, HBS, HODLR, etc), compute a *data-sparse* representation of it. $\rightarrow$ This forms a key component in $O(N)$ solvers for the linear systems resulting from discretization of elliptic PDEs.

Theme: Improve efficiency via *blocking* and *reducing communication.* (And sometimes reducing the asymptotic flop count too!)

Environment 1 — low rank approximation of matrices

Let **A** denote a given $m \times n$ matrix. (We implicitly assume that its singular values decay, so that low-rank approximation makes sense.) Let $\varepsilon > 0$ be a given tolerance.

We then seek factors **Q** and **B** such that

$$\begin{array}{ccccccc} \mathbf{A} & = & \mathbf{Q} & & \mathbf{B} & + & \mathbf{E}, \\ m \times n & & m \times k & & k \times n & & m \times n \end{array}$$

where the error **E** satisfies

$$\|\mathbf{E}\| \leq \varepsilon.$$

We typically also require that **Q** has orthonormal columns.

- Determining a reasonably optimal rank k is part of the problem.
  We assume that k is substantially smaller than $\min(m, n)$.
- Standard software (LAPACK, Matlab, etc) lack built-in functionality for these tasks.
- Objective: Build "shell" algorithms that draw on BLAS3, LAPACK, etc, to solve the low-rank approximation problems efficiently.
- Simple post-processing of the small factor **B** allows the computation of approximate SVD $\mathbf{A} \approx \mathbf{U}_k \, \mathbf{D}_k \, \mathbf{V}_k^*$, and other standard factorizations.

Environment 1 — low rank approximation of matrices

Let **A** denote a given $m \times n$ matrix. (We implicitly assume that its singular values decay, so that low-rank approximation makes sense.) Let $\varepsilon > 0$ be a given tolerance.

We then seek factors **Q** and **B** such that

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{Q}} \quad \underset{k \times n}{\mathbf{B}} + \underset{m \times n}{\mathbf{E}},$$

where the error **E** satisfies

$$\|\mathbf{E}\| \leq \varepsilon.$$

We typically also require that **Q** has orthonormal columns.

References:

- P.G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices". *Applied and Computational Harmonic Analysis*, **30**(1), pp. 47–68, 2011.
  Based on 2006 Yale CS research report YALEU/DCS/RR-1361.
- N. Halko, P.G. Martinsson, J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Review*, **53**(2), pp. 217–288, 2011.
- P.G. Martinsson and S. Voronin, "A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices." *SIAM J. on Scientific Comp.*, **38**(5), S485 – S507, 2016.
- P.G. Martinsson, "Randomized methods for matrix computations." Arxiv.org report #1607.01649, 2018. (To appear in PCMI summer school proceedings.)

An algorithmic template

We build a basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for the column space of $\mathbf{A}$, using a "greedy" algorithm:

(1) $\mathbf{Q}_0 = [\,]$; $\mathbf{B}_0 = [\,]$; $\mathbf{A}_0 = \mathbf{A}$; $j = 0$;
(2) **while** $\|\mathbf{A}_j\| > \varepsilon$
(3) $\quad j = j + 1$
(4) $\quad$ Pick a unit vector $\mathbf{q}_j \in \mathrm{ran}(\mathbf{A}_{j-1})$.
(5) $\quad \mathbf{b}_j = \mathbf{q}_j^* \mathbf{A}_{j-1}$
(6) $\quad \mathbf{Q}_j = [\mathbf{Q}_{j-1}\ \mathbf{q}_j]$
(7) $\quad \mathbf{B}_j = \begin{bmatrix} \mathbf{B}_{j-1} \\ \mathbf{b}_j \end{bmatrix}$
(8) $\quad \mathbf{A}_j = \mathbf{A}_{j-1} - \mathbf{q}_j \mathbf{b}_j$
(9) **end while**

Simple condition: On line (4), pick $\mathbf{q}_j$ as the largest column of $\mathbf{A}_{j-1}$.

Then we recover column-pivoted Gram-Schmidt, which is often an excellent algorithm.

(Round-off errors make some minor modifications necessary; we will discuss this later.)

Problem 1: Hard to *block* efficiently. (Can be done, via, e.g. "tournament pivoting".)

Problem 2: "Typically" gives reasonably close to optimal results, but can be quite bad.

An algorithmic template

We build a basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for the column space of $\mathbf{A}$, using a "greedy" algorithm:

(1) $\mathbf{Q}_0 = [\,]$; $\mathbf{B}_0 = [\,]$; $\mathbf{A}_0 = \mathbf{A}$; $j = 0$;
(2) **while** $\|\mathbf{A}_j\| > \varepsilon$
(3) $\quad j = j + 1$
(4) $\quad$ Pick a unit vector $\mathbf{q}_j \in \mathrm{ran}(\mathbf{A}_{j-1})$.
(5) $\quad \mathbf{b}_j = \mathbf{q}_j^* \mathbf{A}_{j-1}$
(6) $\quad \mathbf{Q}_j = [\mathbf{Q}_{j-1}\ \mathbf{q}_j]$
(7) $\quad \mathbf{B}_j = \begin{bmatrix} \mathbf{B}_{j-1} \\ \mathbf{b}_j \end{bmatrix}$
(8) $\quad \mathbf{A}_j = \mathbf{A}_{j-1} - \mathbf{q}_j \mathbf{b}_j$
(9) **end while**

Optimal condition: On line (4), pick $\mathbf{q}_j$ as a *minimizer* of

$$\min_{\|\mathbf{q}\|=1} \|\mathbf{A}_{j-1} - \mathbf{q}\mathbf{q}^* \mathbf{A}_{j-1}\|.$$

Problem: Computationally hard to find the minimizer.

An algorithmic template — now *randomized*

We build a basis $\{\mathbf{q}_j\}_{j=1}^{k}$ for the column space of $\mathbf{A}$, using a "greedy" algorithm:

(1) $\mathbf{Q}_0 = [\,]$; $\mathbf{B}_0 = [\,]$; $\mathbf{A}_0 = \mathbf{A}$; $j = 0$;

(2) **while** $\|\mathbf{A}_j\| > \varepsilon$

(3) $\quad j = j + 1$

(4a) $\quad$ Draw a random vector ω whose entries are iid Gaussian random variables.

(4b) $\quad$ Set $\mathbf{y} = \mathbf{A}_{j-1}\omega$.

(4c) $\quad$ Normalize so that $\mathbf{q}_j = \frac{1}{\|\mathbf{y}\|}\,\mathbf{y}$.

(5) $\quad \mathbf{b}_j = \mathbf{q}_j^* \mathbf{A}_{j-1}$

(6) $\quad \mathbf{Q}_j = [\mathbf{Q}_{j-1}\ \mathbf{q}_j]$

(7) $\quad \mathbf{B}_j = \begin{bmatrix} \mathbf{B}_{j-1} \\ \mathbf{b}_j \end{bmatrix}$

(8) $\quad \mathbf{A}_j = \mathbf{A}_{j-1} - \mathbf{q}_j \mathbf{b}_j$

(9) **end while**

Simple to implement.

Often reasonably close to optimal.

Very easy to block.

An algorithmic template — now randomized and *blocked*

Pick a "block size" ℓ.

(1) $\mathbf{Q} = [\,]$; $\mathbf{B} = [\,]$;

(2) **while** $\|\mathbf{A}\| > \varepsilon$

(3) Draw an $n \times \ell$ random matrix $\mathbf{R}$.

(4) Compute the $m \times \ell$ matrix $\mathbf{Q}_{\text{new}} = \texttt{qr}(\mathbf{AR}, 0)$.

(5) $\mathbf{B}_{\text{new}} = \mathbf{Q}_{\text{new}}^{*}\mathbf{A}$

(6) $\mathbf{Q} = [\mathbf{Q}\ \mathbf{Q}_{\text{new}}]$

(7) $\mathbf{B} = \begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\text{new}} \end{bmatrix}$

(8) $\mathbf{A} = \mathbf{A} - \mathbf{Q}_{\text{new}}\mathbf{B}_{\text{new}}$

(9) **end while**

The scheme presented works very well for matrices whose singular values decay rapidly.

Note that every line can be executed by BLAS3, except (4) for which we use LAPACK.

When the singular values do not decay rapidly, we apply a *power* of $\mathbf{A}$.

An algorithmic template — now randomized, blocked, and *accuracy-enhanced*

Pick a "block size" ℓ, and a small integer q, say $q = 1$, or $q = 2$.

(1) $\mathbf{Q} = [\,]$; $\mathbf{B} = [\,]$;

(2) **while** $\|\mathbf{A}\| > \varepsilon$

(3) Draw an $n \times \ell$ random matrix $\mathbf{R}$.

(4) Compute the $m \times \ell$ matrix $\mathbf{Q}_{\text{new}} = \texttt{qr}((\mathbf{A}\mathbf{A}^*)^q\mathbf{A}\mathbf{R}, 0)$.

(5) $\mathbf{B}_{\text{new}} = \mathbf{Q}_{\text{new}}^*\mathbf{A}$

(6) $\mathbf{Q} = [\mathbf{Q}\ \mathbf{Q}_{\text{new}}]$

(7) $\mathbf{B} = \begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\text{new}} \end{bmatrix}$

(8) $\mathbf{A} = \mathbf{A} - \mathbf{Q}_{\text{new}}\mathbf{B}_{\text{new}}$

(9) **end while**

The only thing remaining is to deal with loss of orthonormality due to round-off errors.

An algorithmic template — now randomized, blocked, and accuracy-enhanced

Pick a "block size" ℓ, and a small integer q, say $q = 0$, $q = 1$, or $q = 2$.

THE "RAND-QB" ALGORITHM

(1) $\mathbf{Q} = [\,]$; $\mathbf{B} = [\,]$;

(2) **while** $\|\mathbf{A}\| > \varepsilon$

(3) Draw an $n \times \ell$ random matrix $\mathbf{R}$.

(4a) Compute the $m \times \ell$ matrix $\mathbf{Y} = \mathrm{qr}((\mathbf{A}\mathbf{A}^*)^q\mathbf{A}\mathbf{R}, 0)$.

(4b) Reproject $\mathbf{Y}$ away from the range of $\mathbf{Q}$: $\mathbf{Y} = \mathbf{Y} - \mathbf{Q}(\mathbf{Q}^*\mathbf{Y})$.

(4c) Compute the $m \times \ell$ matrix $\mathbf{Q}_{\mathrm{new}} = \mathrm{qr}(\mathbf{Y}, 0)$.

(5) $\mathbf{B}_{\mathrm{new}} = \mathbf{Q}_{\mathrm{new}}^*\mathbf{A}$

(6) $\mathbf{Q} = [\mathbf{Q}\ \mathbf{Q}_{\mathrm{new}}]$

(7) $\mathbf{B} = \begin{bmatrix} \mathbf{B} \\ \mathbf{B}_{\mathrm{new}} \end{bmatrix}$

(8) $\mathbf{A} = \mathbf{A} - \mathbf{Q}_{\mathrm{new}}\mathbf{B}_{\mathrm{new}}$

(9) **end while**

With minor modifications, we can avoid updating $\mathbf{A}$. This is crucial for sparse matrices, and in situations where we can only access $\mathbf{A}$ via its action on vectors.

Given the "QB-factorization," standard factorizations can easily be computed

Suppose that you have an approximate factorization

$$\mathbf{A} = \mathbf{QB} + \mathbf{E},$$

where $\mathbf{Q}$ is orthonormal and $\|\mathbf{E}\|$ is small.

How to get an approximate SVD: Perform a full SVD of the small matrix $\mathbf{B}$:

$$[\hat{\mathbf{U}}, \mathbf{D}, \mathbf{V}] = \texttt{svd}(\mathbf{B}, \texttt{'econ'}).$$

Then simply set $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$ and you will get a partial SVD

$$\mathbf{A} = \mathbf{QB} + \mathbf{E} = \mathbf{QUDV}^* + \mathbf{E} = \mathbf{UDV}^* + \mathbf{E}.$$

Note that the error is exactly the same as the error in the QB.

Given the "QB-factorization," standard factorizations can easily be computed

Suppose that you have an approximate factorization

$$\mathbf{A} = \mathbf{QB} + \mathbf{E},$$

where $\mathbf{Q}$ is orthonormal and $\|\mathbf{E}\|$ is small.

How to get an approximate SVD: Perform a full SVD of the small matrix $\mathbf{B}$:

$$[\hat{\mathbf{U}}, \mathbf{D}, \mathbf{V}] = \texttt{svd}(\mathbf{B}, \texttt{'econ'}).$$

Then simply set $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$ and you will get a partial SVD

$$\mathbf{A} = \mathbf{QB} + \mathbf{E} = \mathbf{QUDV}^* + \mathbf{E} = \mathbf{UDV}^* + \mathbf{E}.$$

Note that the error is exactly the same as the error in the QB.

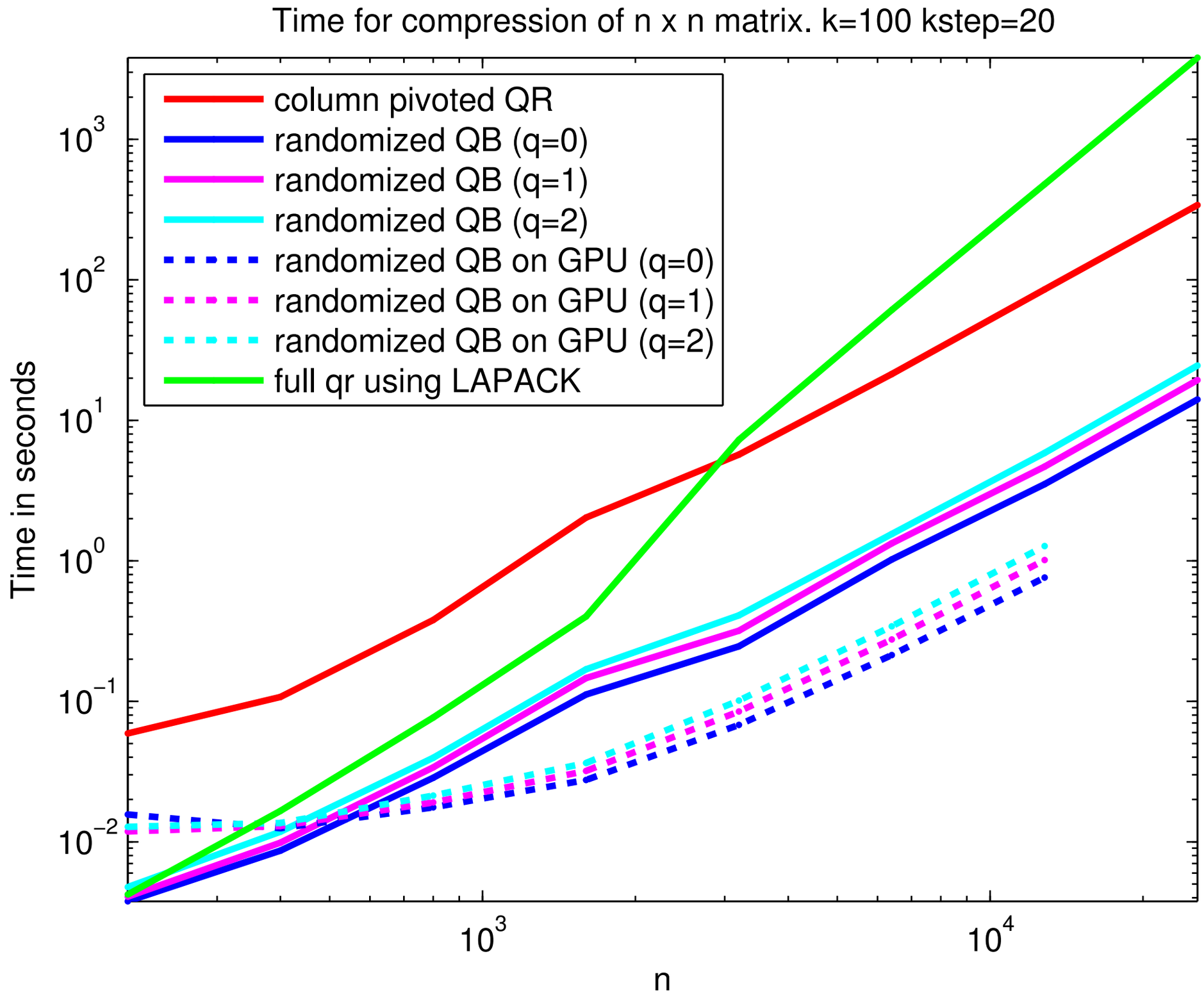
How to get an approximate QR: Perform a full QR of the small matrix $\mathbf{B}$:

$$[\hat{\mathbf{Q}}, \mathbf{R}, \mathbf{P}] = \texttt{qr}(\mathbf{B}, 0).$$

Then simply set $\tilde{\mathbf{Q}} = \mathbf{Q}\hat{\mathbf{Q}}$ and you will get a partial QR

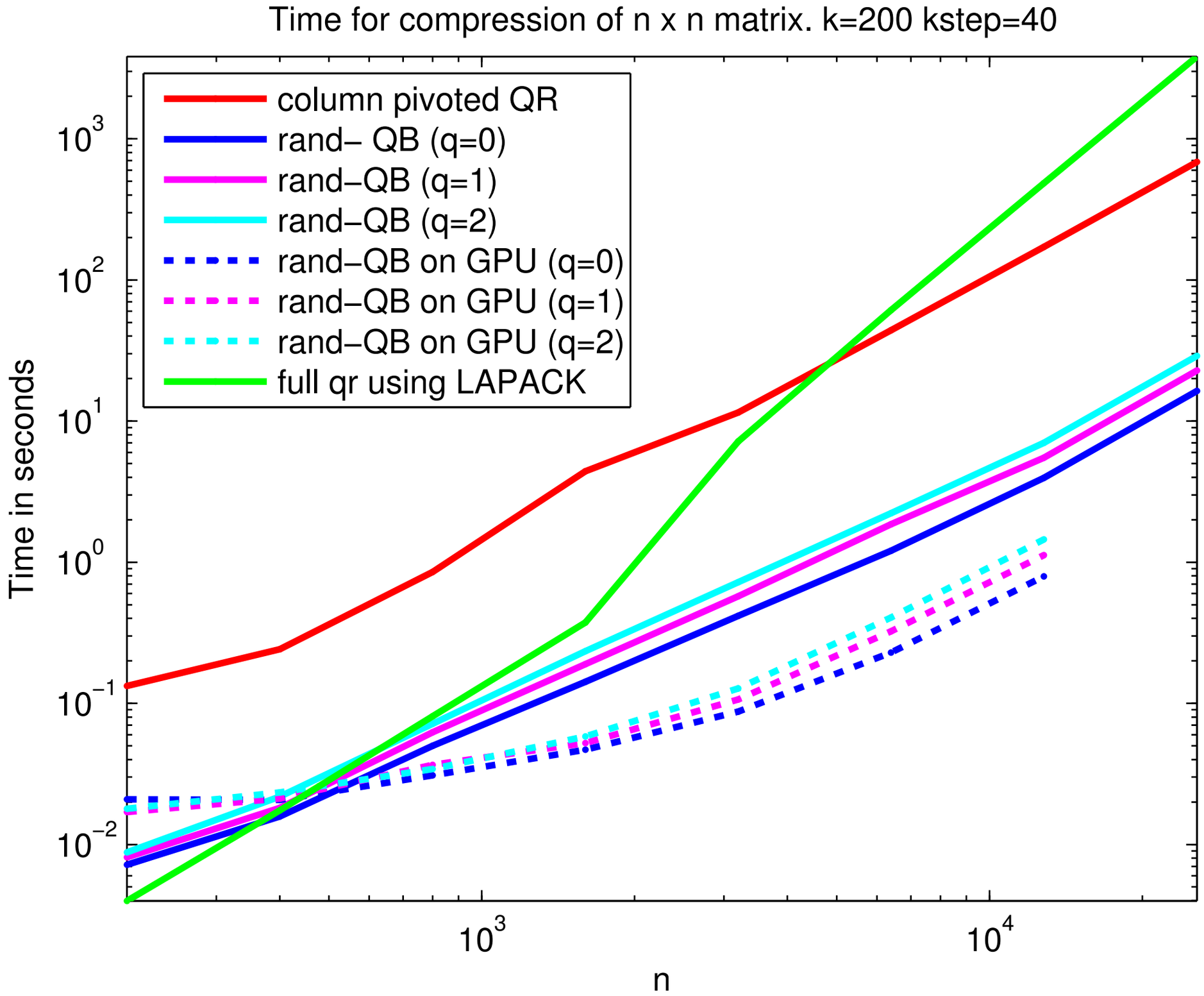
$$\mathbf{AP} = \mathbf{QBP} + \mathbf{EP} = \mathbf{Q}\hat{\mathbf{Q}}\mathbf{R} + \mathbf{EP} = \tilde{\mathbf{Q}}\mathbf{R} + \mathbf{EP}.$$

The error is exactly the same as the error in the QB (modulo column permutations).



Everything is implemented in Matlab. The “full qr” line refers to Matlab built in qr.

Caveat: Matlab overhead makes column-pivoted QR slower than it could be.



Everything is implemented in Matlab. The “full qr” line refers to Matlab built in qr.

Caveat: Matlab overhead makes column-pivoted QR slower than it could be.

Example: Accuracy for synthetic matrix with rapidly decaying spectrum

Consider a matrix *defined* by its 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 $r = \min(m, n)$, where **U** and **V** are random orthonormal matrices, and

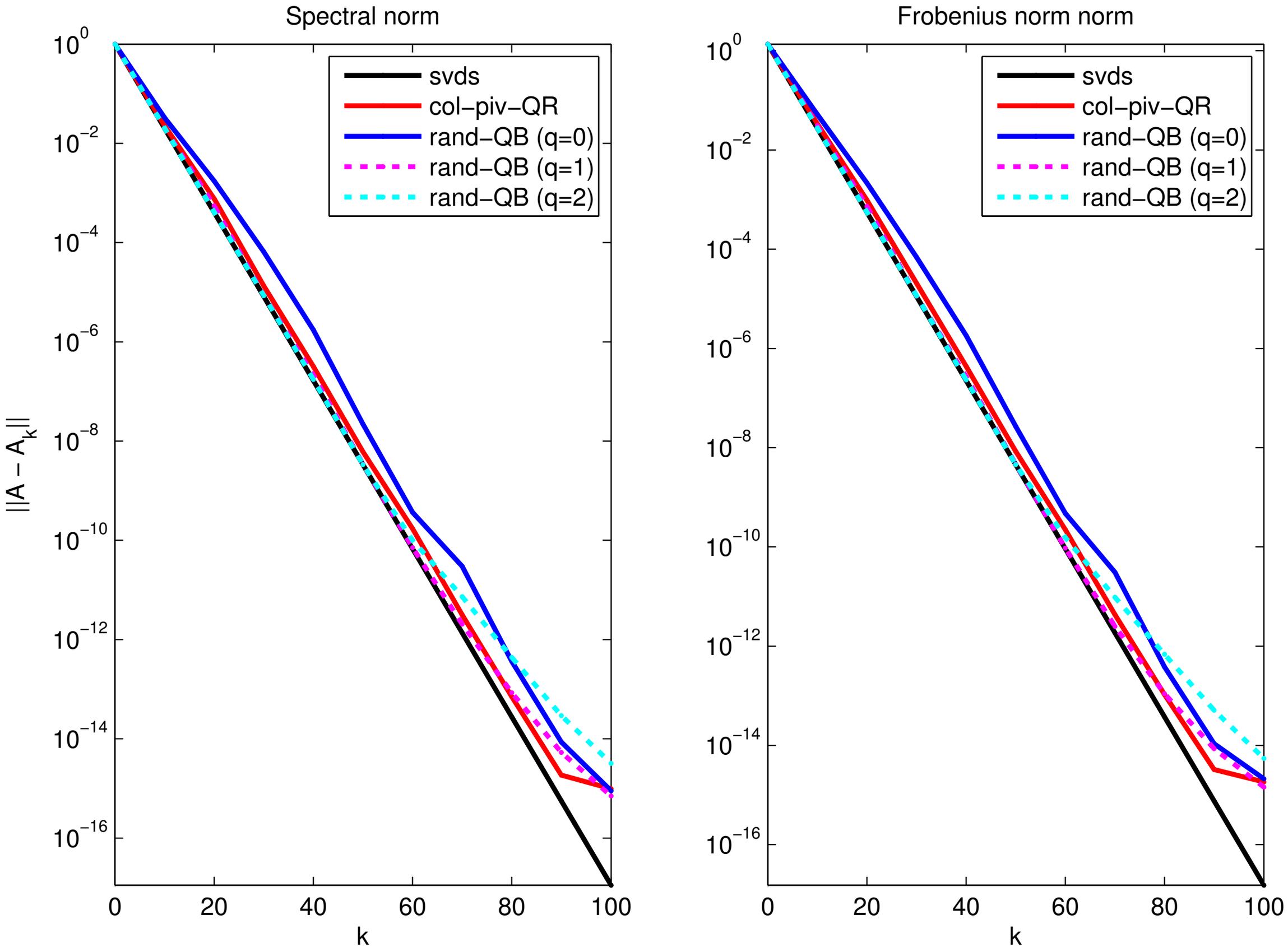
$$\mathbf{D} = \mathtt{diag}(1, \alpha, \alpha^2, \alpha^3, \ldots),$$

where α is chosen so that

$$\alpha^{90} = 10^{-15}.$$

In this example, $n = 400$.

Error $\|\mathbf{A} - \mathbf{A}_k\|$ *for the blocked* ($\ell = 20$) *version* ($\mathbf{A}$ *is* 400×400)



Example: Accuracy for synthetic matrix with slowly decaying spectrum

Consider a matrix *defined* by its 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 $r = \min(m, n)$, where **U** and **V** are random orthonormal matrices, and

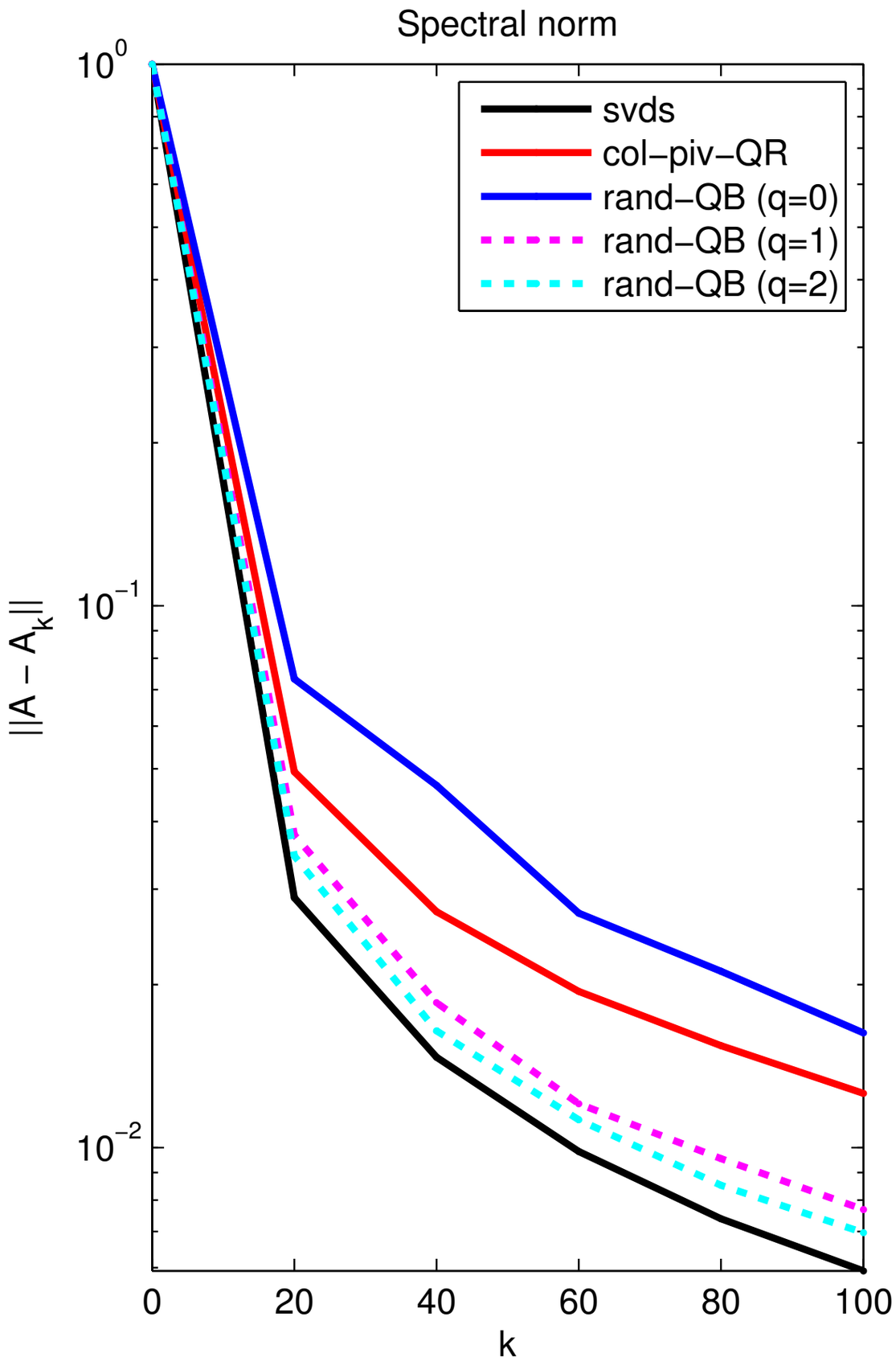
$$\mathbf{D} = \mathtt{diag}(\sigma_1, \sigma_2, \sigma_3, \ldots),$$

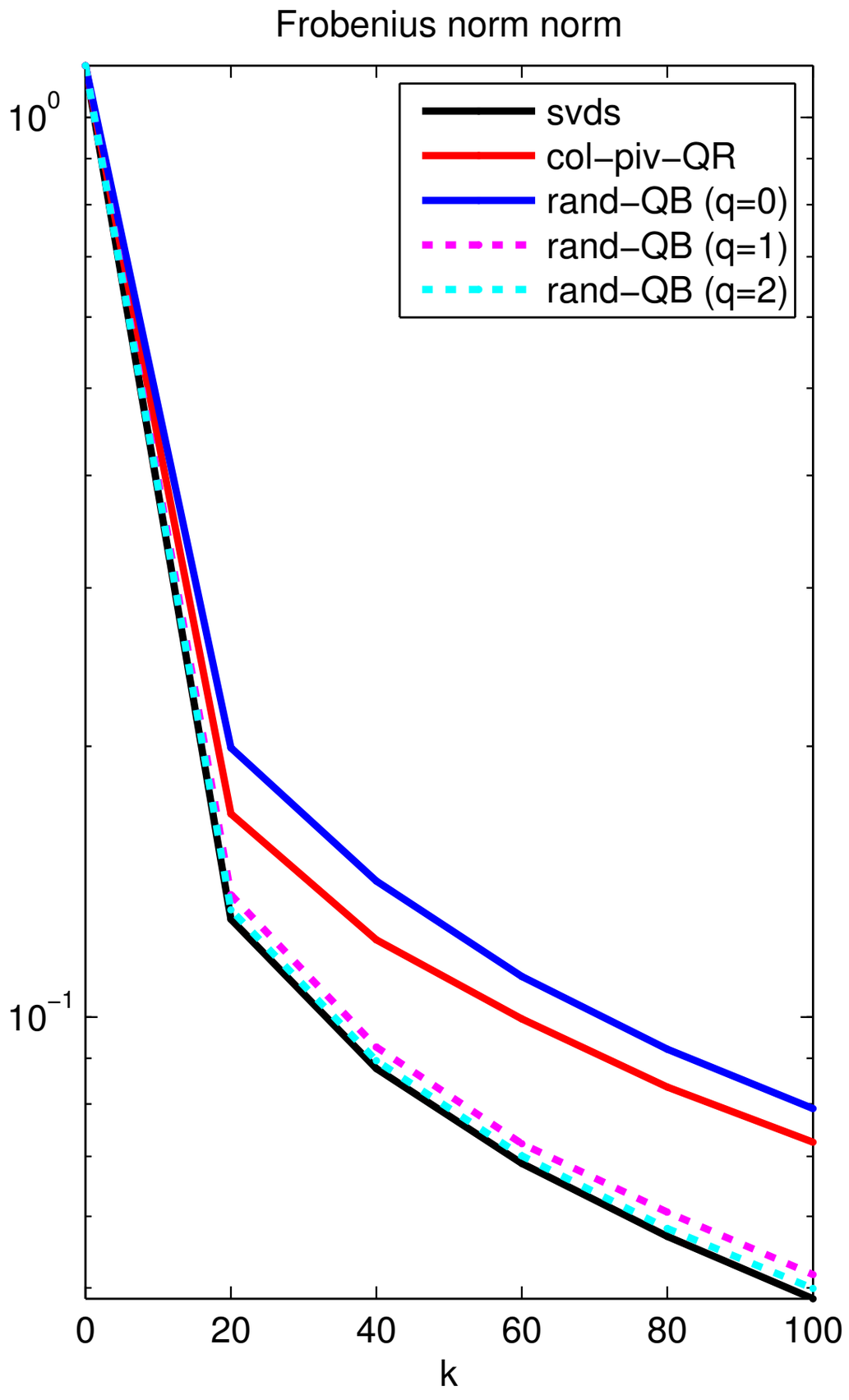
where

$$\sigma_j = \frac{1}{\sqrt{1 + 3\,(j-1)}}.$$

In this example, $m = 500$ and $n = 300$.

Error $\|\mathbf{A} - \mathbf{A}_k\|$ *for the blocked* ($\ell = 20$) *version* ($\mathbf{A}$ *is* 500×300)





Randomized low-rank approximation of matrices

The methods described so far are *very* easy to implement. Can be done in Matlab, or in C/Fortran using standard subroutines (`dgemm`, `dgeqrf`).

Computational speed is good; in particular on GPUs.

The accuracy is very good. With two sweeps of power iteration ($q = 2$), it compares very favorably to column-pivoted QR, and is almost as good as SVD.

Randomized low-rank approximation of matrices

The methods described so far are *very* easy to implement. Can be done in Matlab, or in C/Fortran using standard subroutines (`dgemm`, `dgeqrf`).

Computational speed is good; in particular on GPUs.

The accuracy is very good. With two sweeps of power iteration ($q = 2$), it compares very favorably to column-pivoted QR, and is almost as good as SVD.

Caveat: This is efficient only when the rank k is small, $k \ll \min(m, n)$.

Question: Can we device a method that works well *for any rank?*

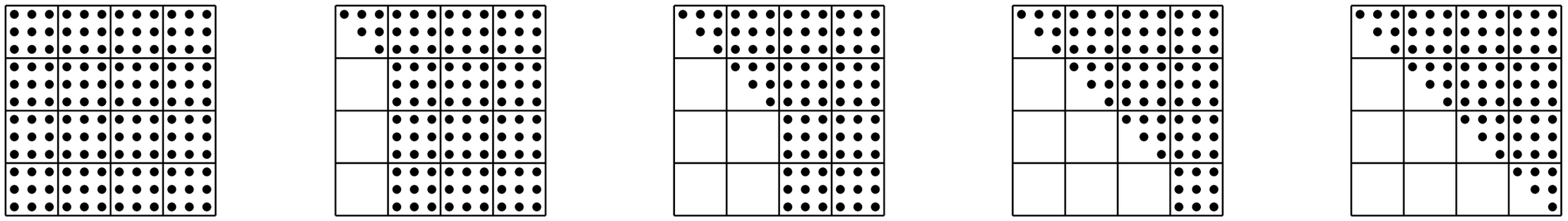
Environment 2: Computing a traditional QR factorization

Given a dense $m \times n$ matrix $\mathbf{A}$ (with $m \geq n$) compute QR (or RRQR)

$$\begin{array}{cccc} \mathbf{A} & \mathbf{P} & \approx & \mathbf{Q} \quad \mathbf{R} \\ m \times n & n \times n & & m \times k \;\; k \times n \end{array}$$

for either $k = n$ (full factorization) or k comparable to $\min(m, n)$. 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:



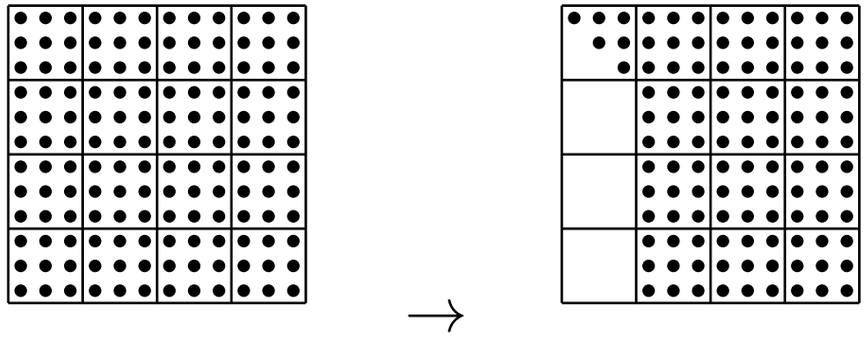
$$\mathbf{A}_0 = \mathbf{A} \qquad \mathbf{A}_1 = \mathbf{Q}_1^* \mathbf{A}_0 \mathbf{P}_1 \qquad \mathbf{A}_2 = \mathbf{Q}_2^* \mathbf{A}_1 \mathbf{P}_2 \qquad \mathbf{A}_3 = \mathbf{Q}_3^* \mathbf{A}_2 \mathbf{P}_3 \qquad \mathbf{A}_4 = \mathbf{Q}_4^* \mathbf{A}_3 \mathbf{P}_4$$

Each $\mathbf{Q}_j$ is a product of Householder reflectors. Each $\mathbf{P}_j$ is a permutation matrix computed via randomized sampling.

Environment 2: Computing a traditional QR factorization

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 **G** of size $b \times m$ and form

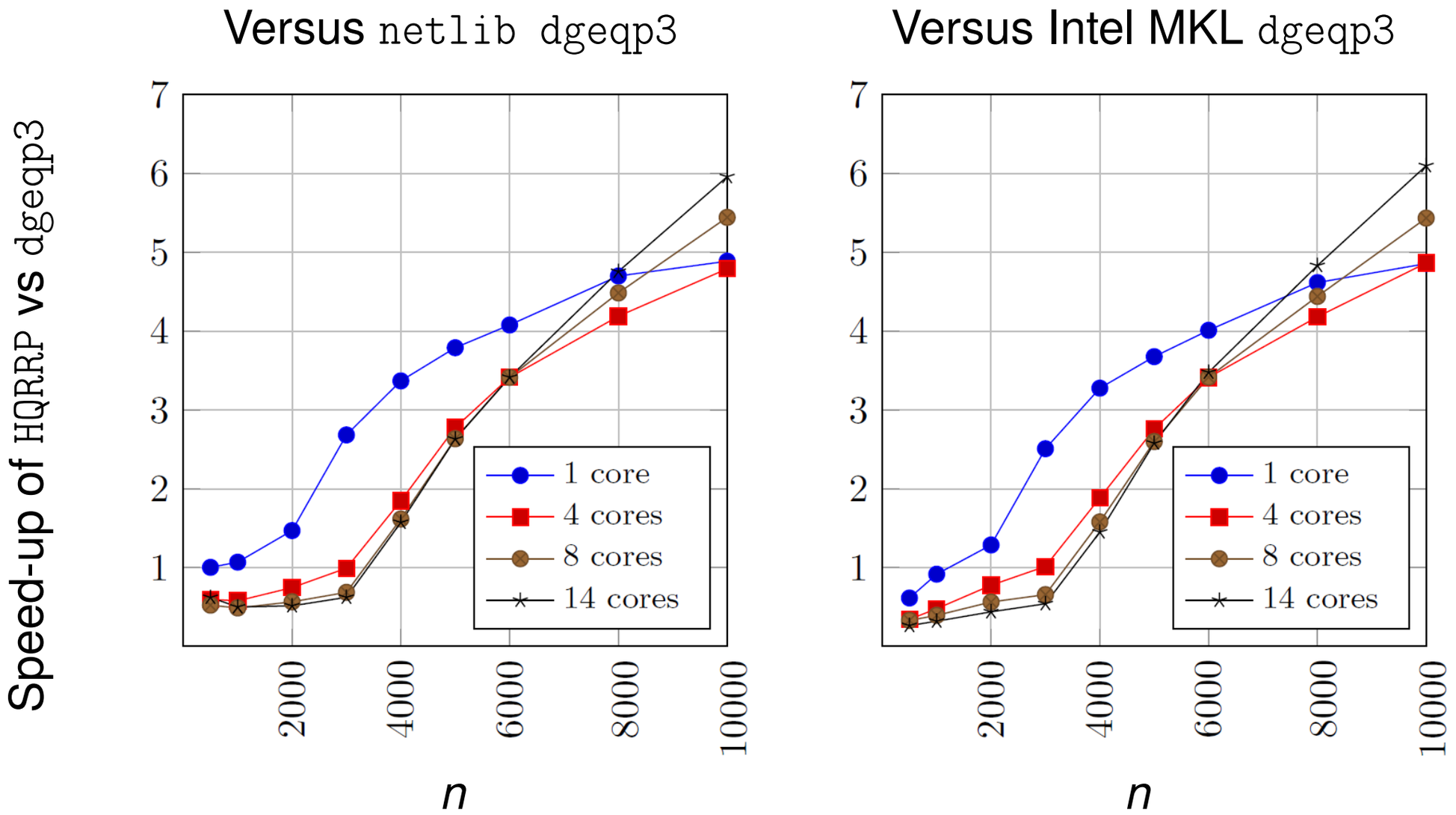
$$\underset{b \times n}{\mathbf{Y}} = \underset{b \times m}{\mathbf{G}} \; \underset{m \times n}{\mathbf{A}}$$

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**:

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

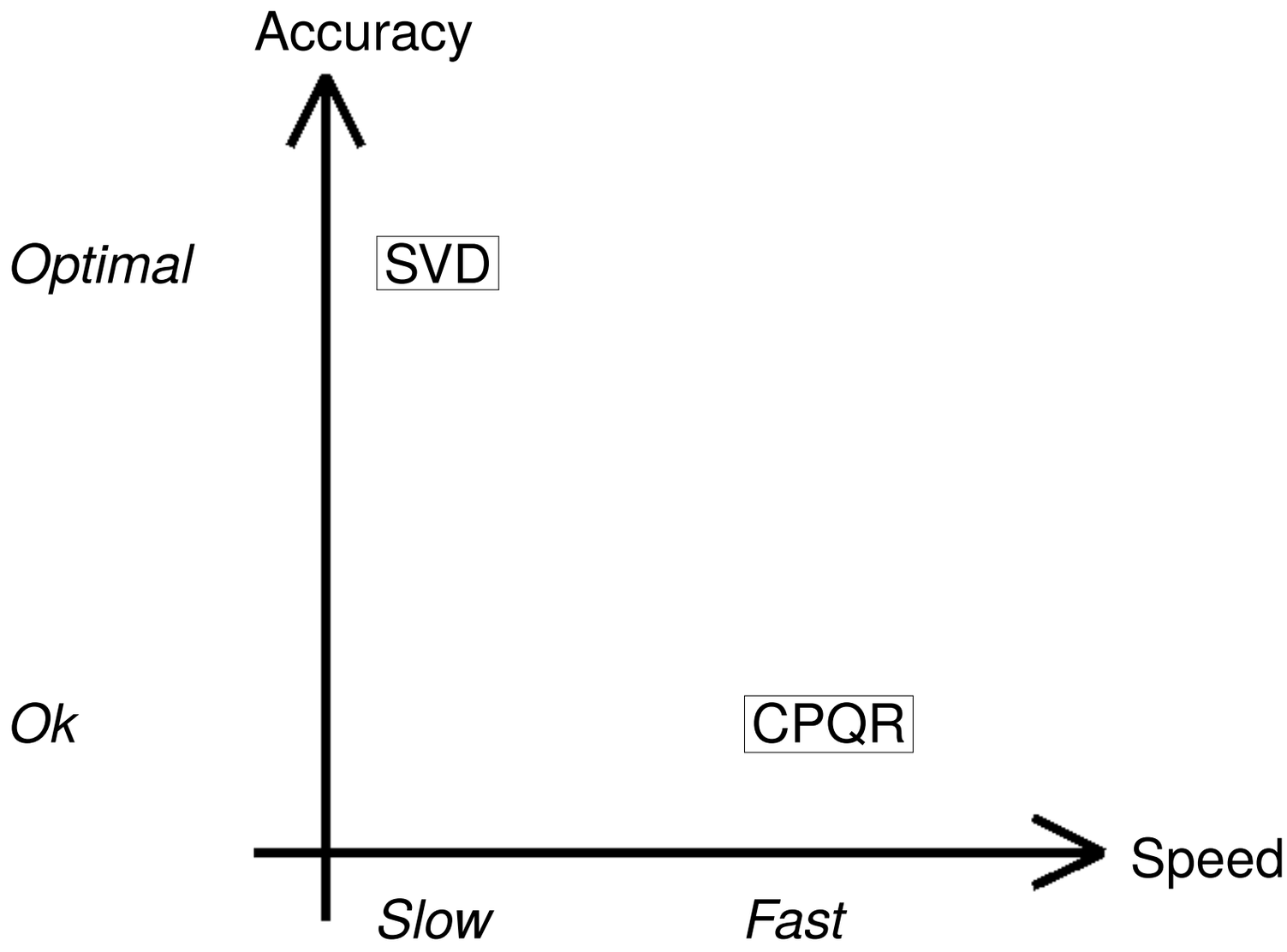
Environment 2: Computing a traditional QR factorization



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/`

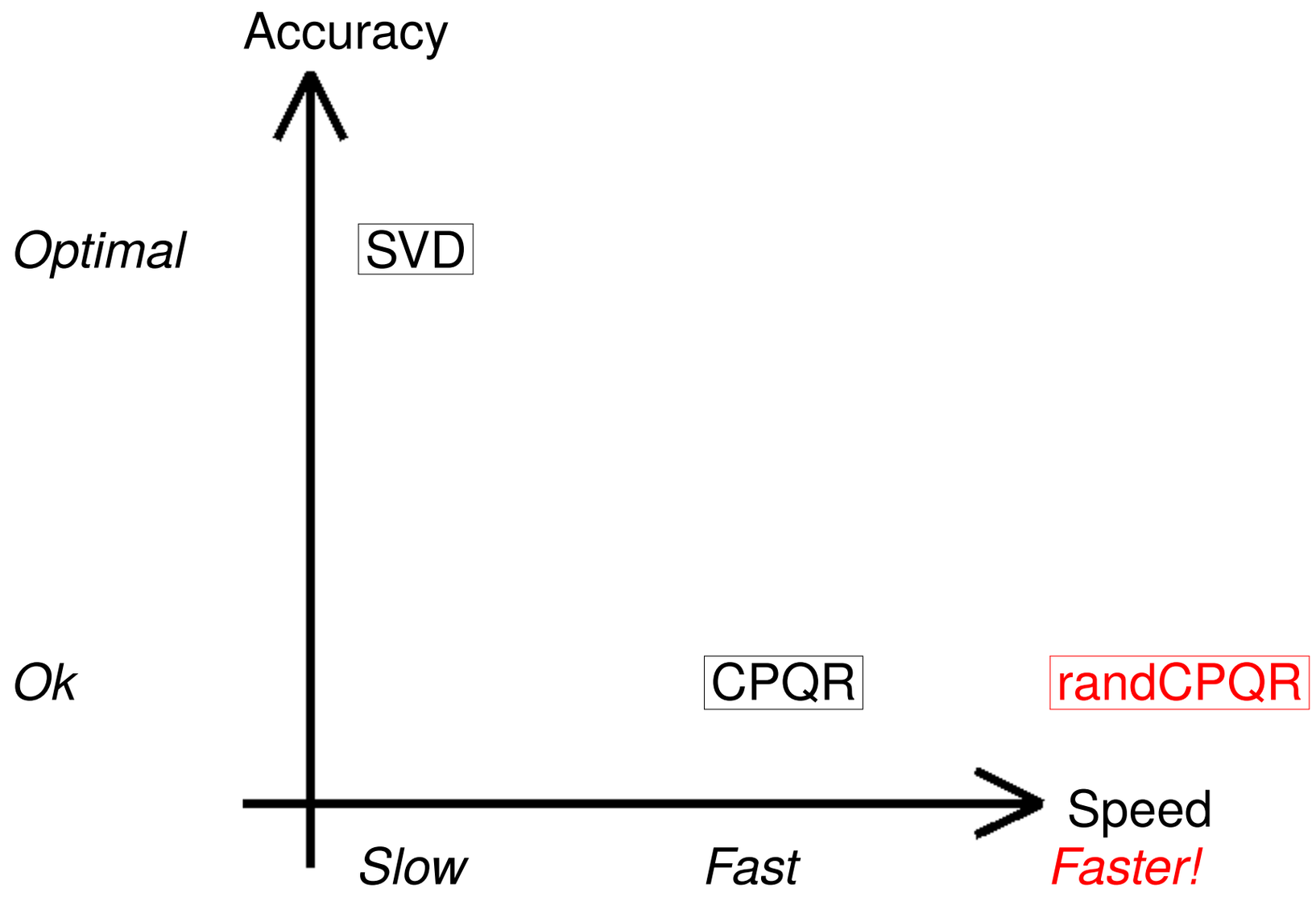
Environment 2: Computing a traditional QR factorization

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:



Environment 2: Computing a traditional QR factorization

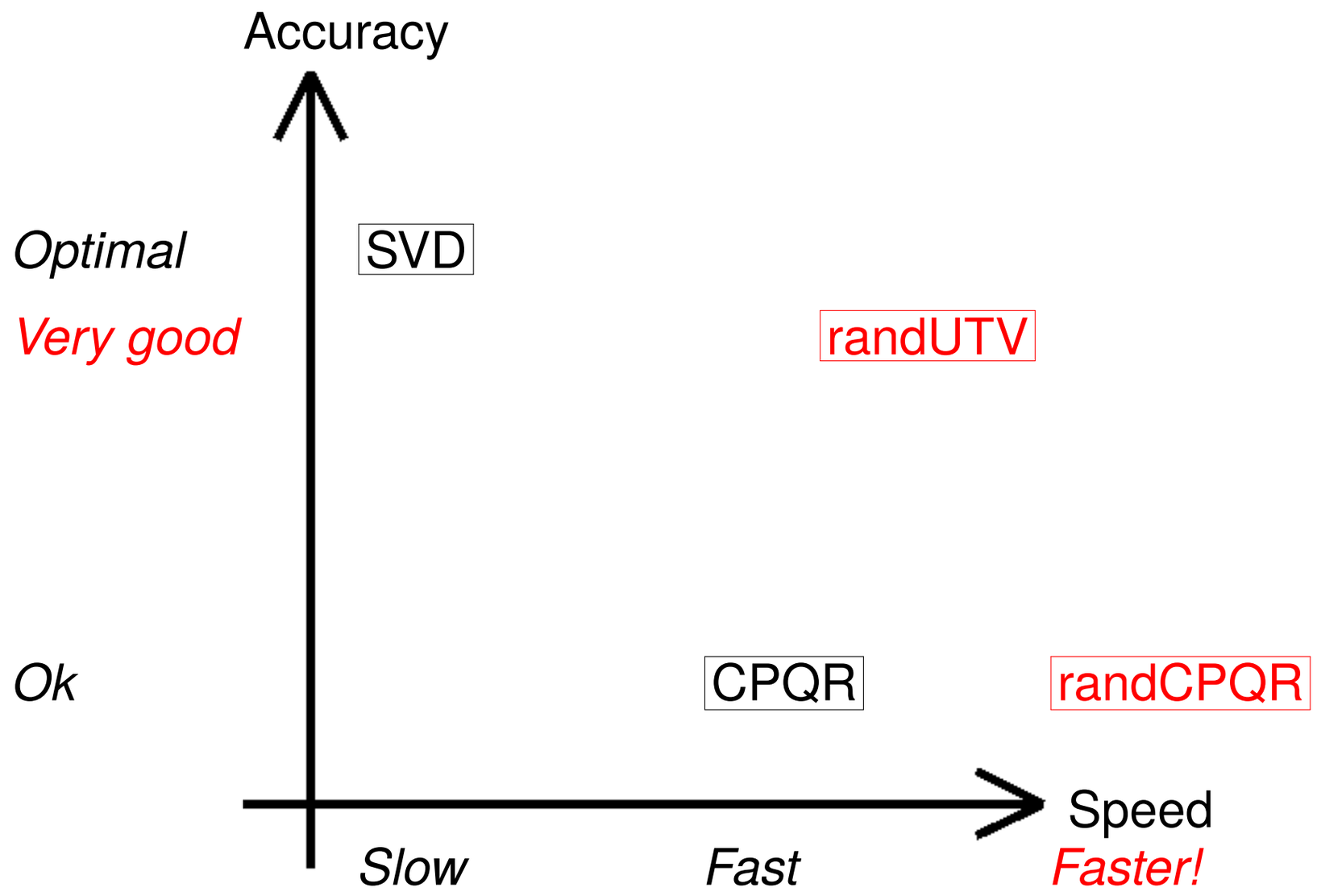
For the task of computing low-rank approximations to matrices, the classical choice is between SVD and column pivoted QR (CPQR). SVD is slow, and CPQR is inaccurate:



Randomized CPQR is faster than CPQR, but is no better in terms of accuracy.

Environment 2: Computing a traditional QR factorization

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:



Randomized CPQR is faster than CPQR, but is no better in terms of accuracy.

Randomized UTV is faster than CPQR, and attains very close to SVD accuracy!

Additionally, randUTV parallelizes well and supports partial factorization.

Environment 2: Accelerate FULL factorizations of matrices

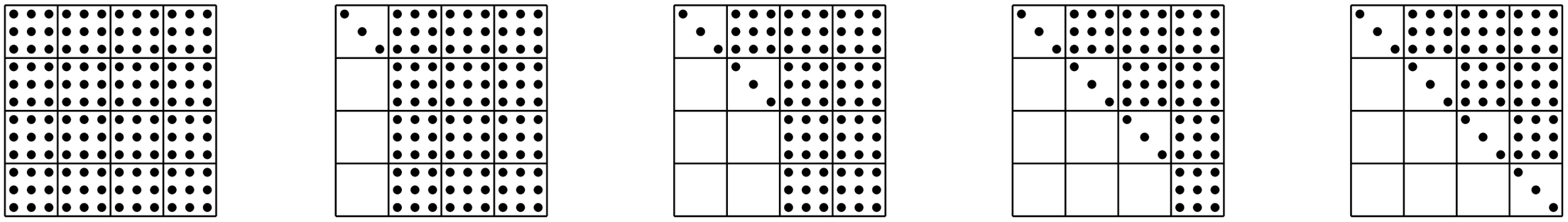
Given a dense $n \times n$ matrix $\mathbf{A}$, compute a factorization

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

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:

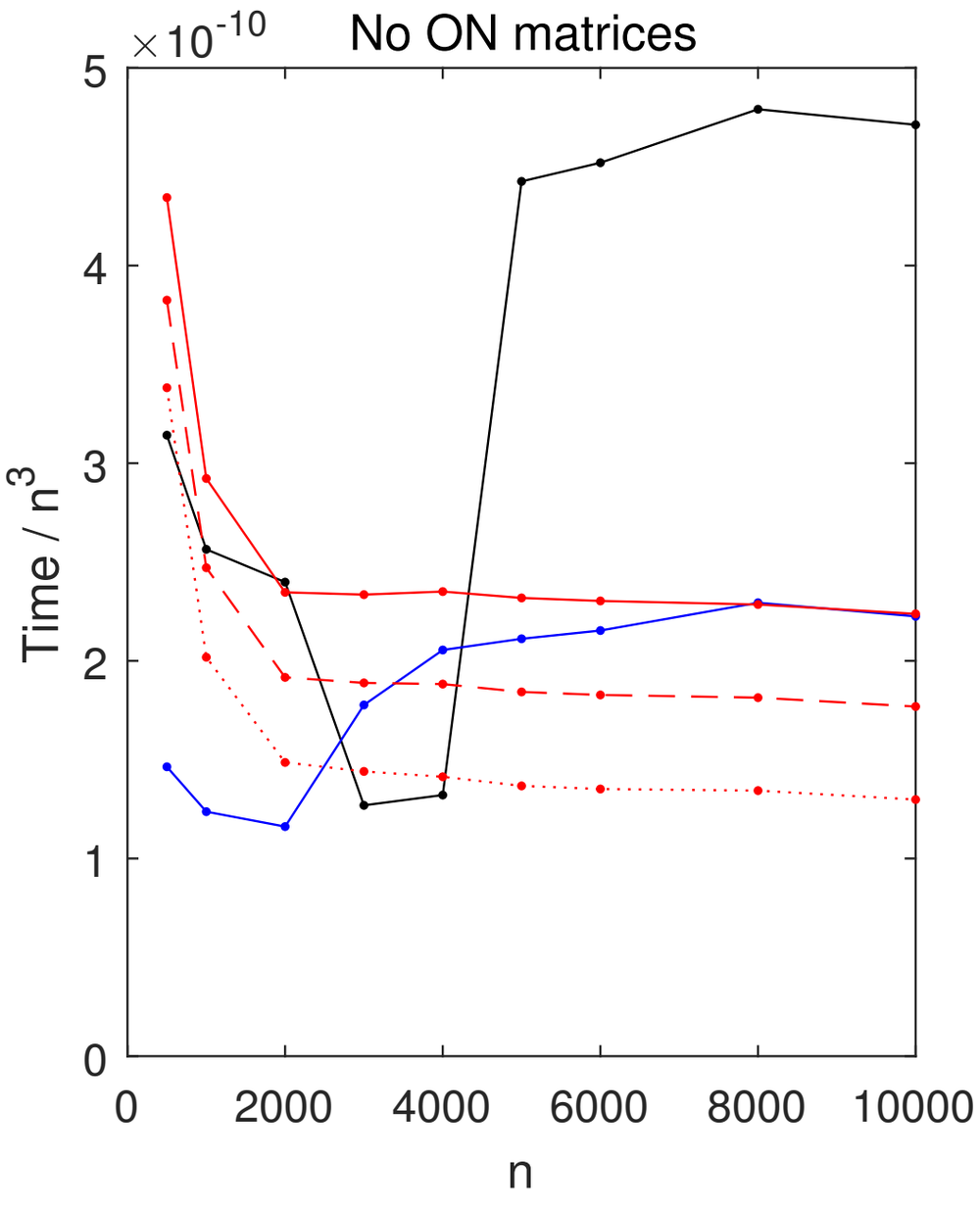


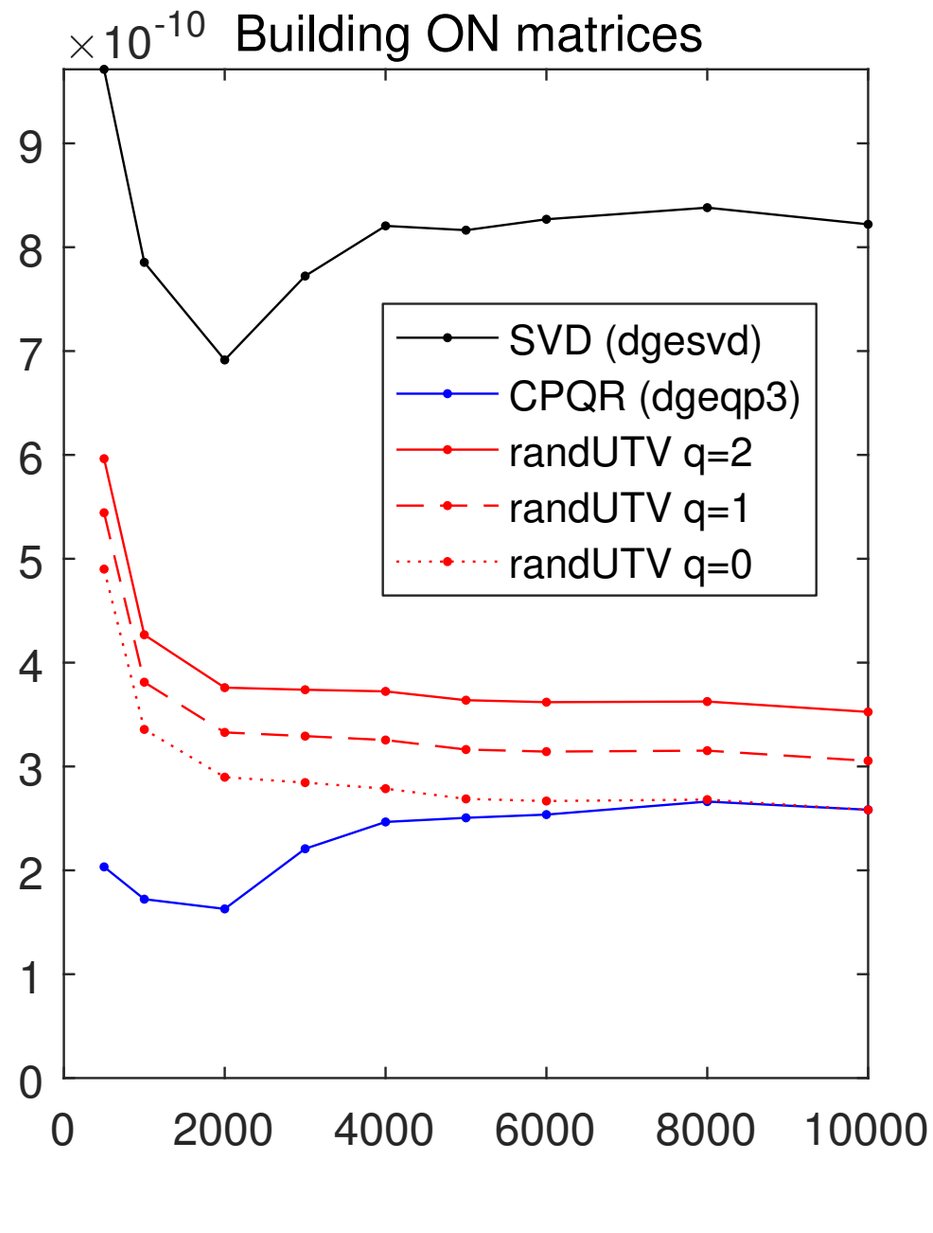
$$\mathbf{A}_0 = \mathbf{A} \qquad \mathbf{A}_1 = \mathbf{U}_1^*\mathbf{A}_0\mathbf{V}_1 \qquad \mathbf{A}_2 = \mathbf{U}_2^*\mathbf{A}_1\mathbf{V}_2 \qquad \mathbf{A}_3 = \mathbf{U}_3^*\mathbf{A}_2\mathbf{V}_3 \qquad \mathbf{A}_4 = \mathbf{U}_4^*\mathbf{A}_3\mathbf{V}_4$$

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.

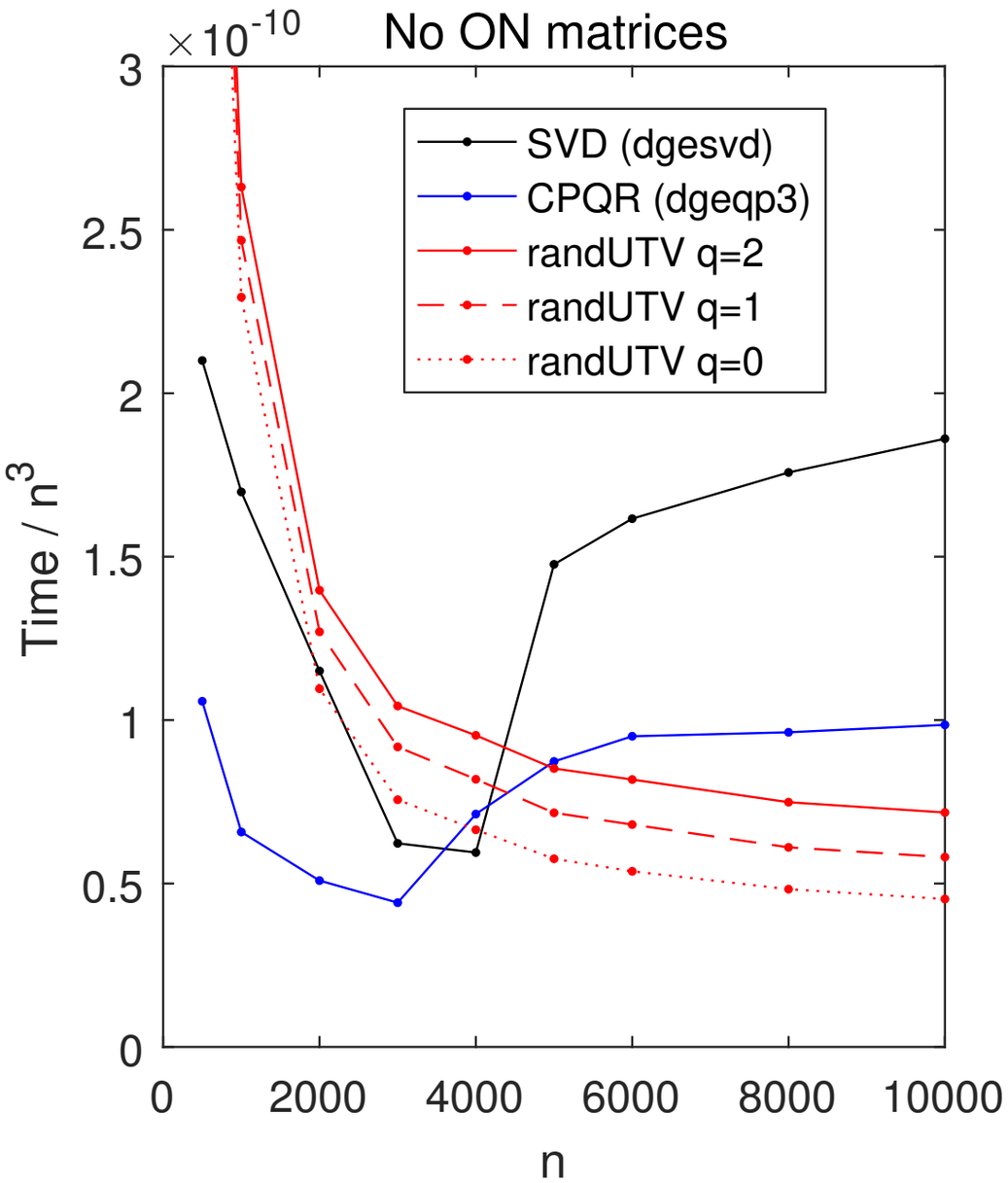
Computational speed of randUTV — 1 core

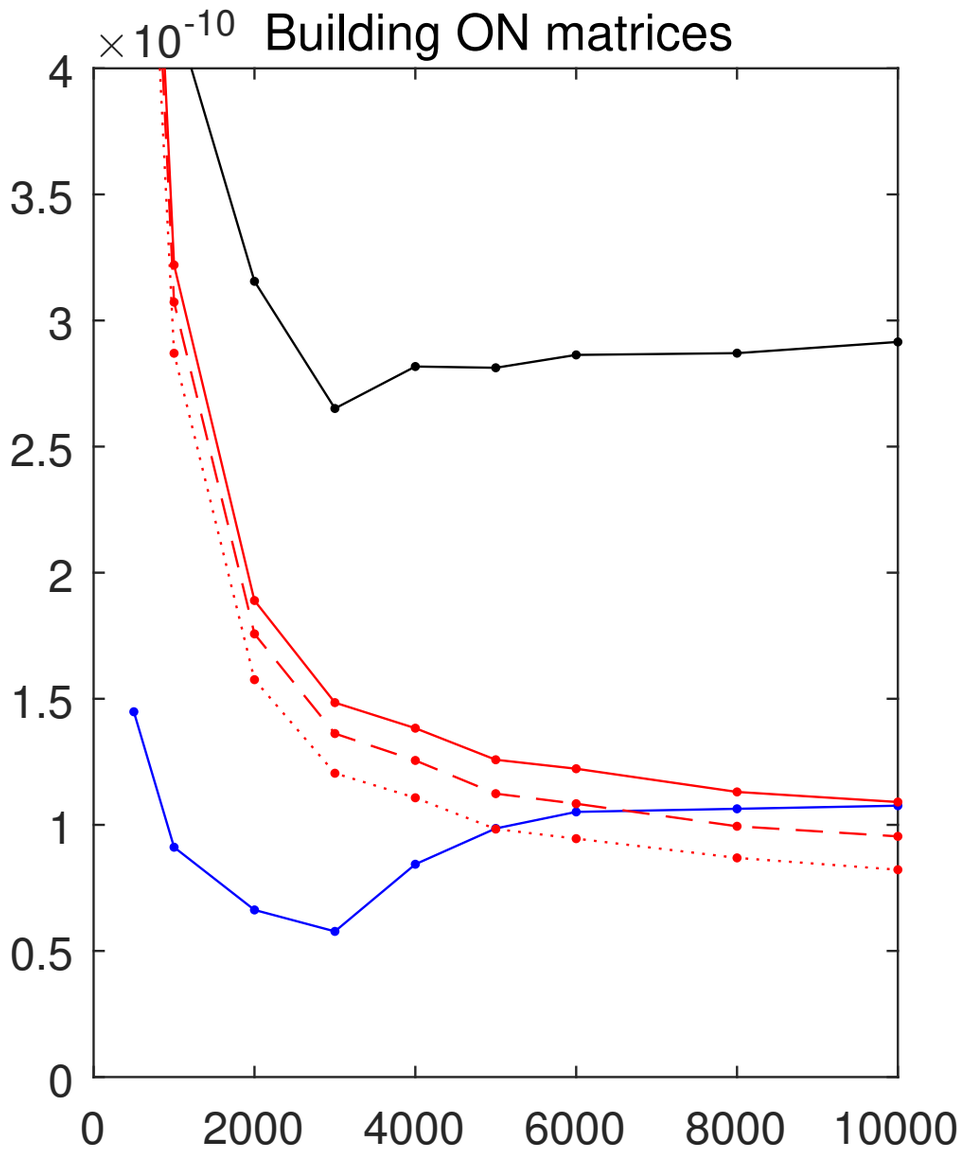




Computational speed of randUTV (red) compared to SVD (black) and column pivoted QR (blue). Each run is for a full factorization of an $n \times n$ matrix. We compare against Intel MKL LAPACK. Observe that the vertical axis is execution time scaled by n^3!

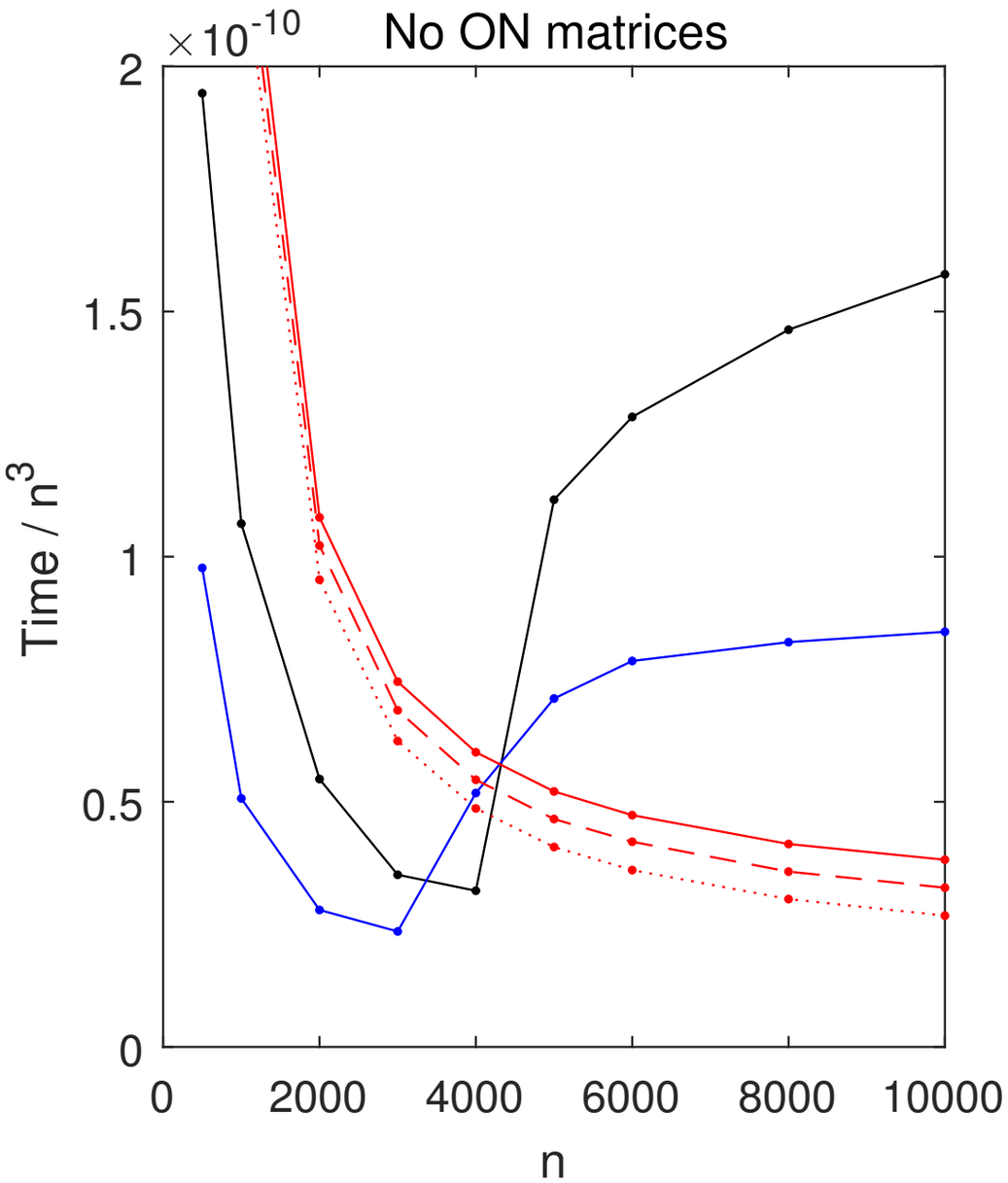
Computational speed of randUTV — 4 cores

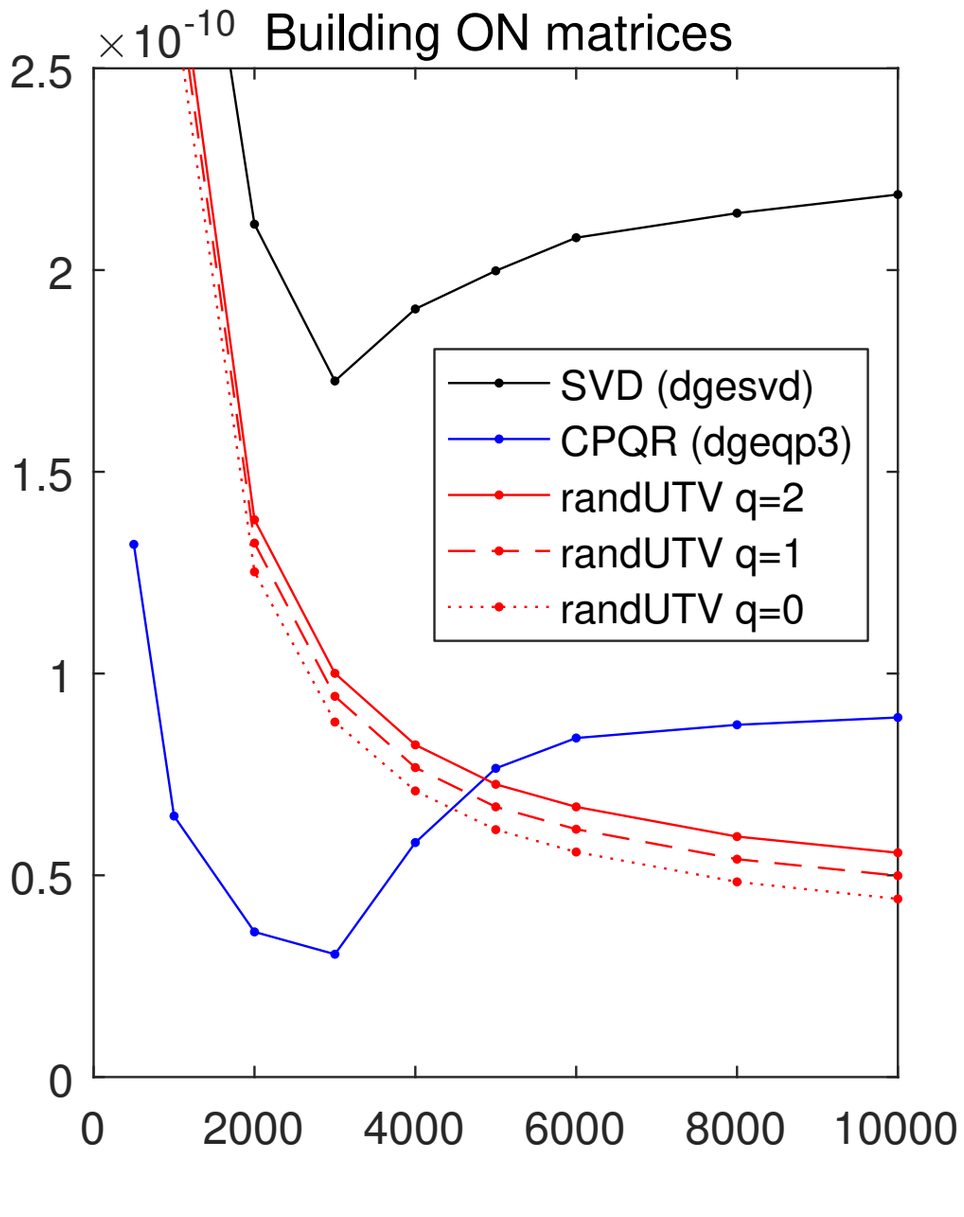




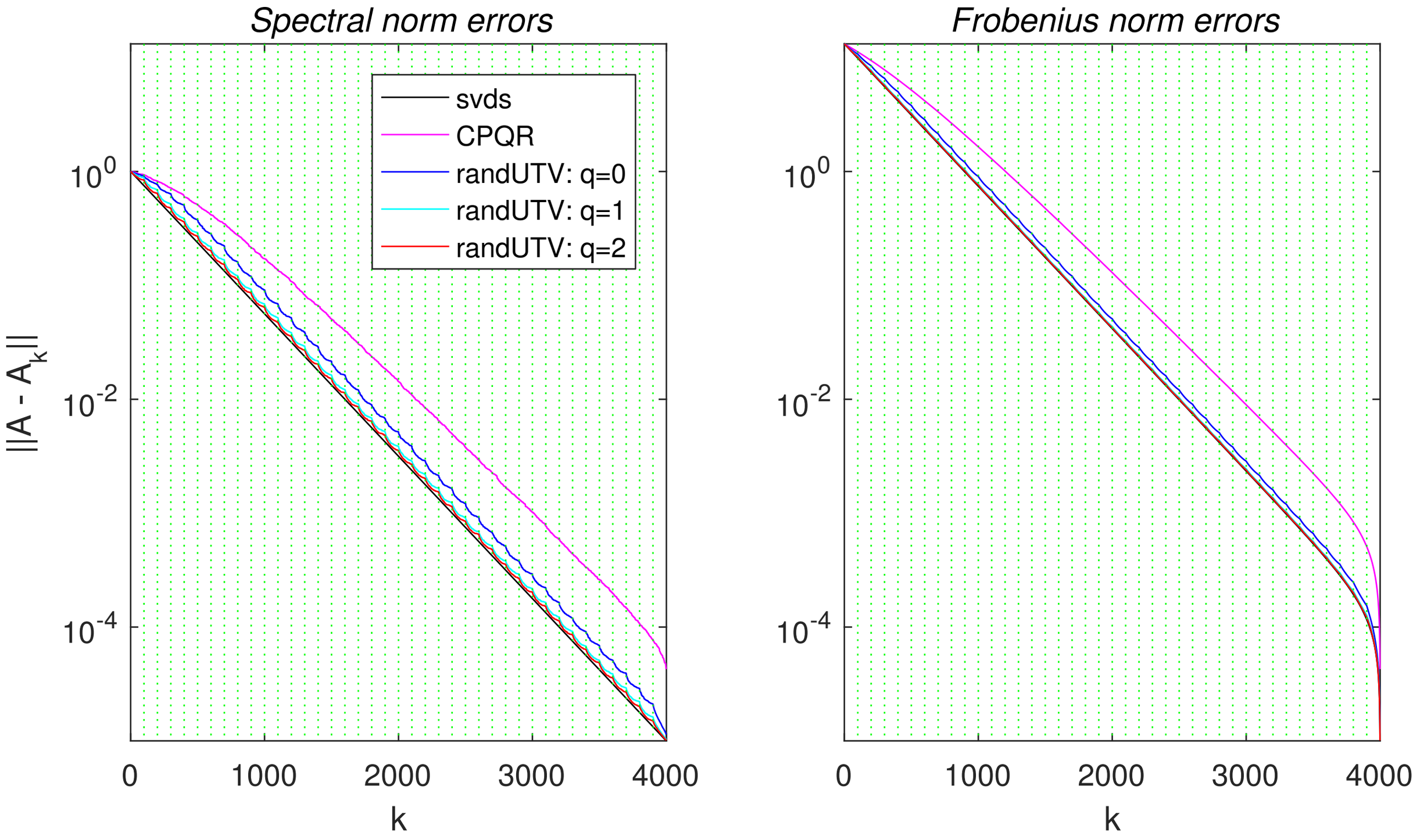
Computational speed of randUTV (red) compared to SVD (black) and column pivoted QR (blue). Each run is for a full factorization of an $n \times n$ matrix. We compare against Intel MKL LAPACK. Observe that the vertical axis is execution time scaled by n^3!

Computational speed of randUTV — 12 cores

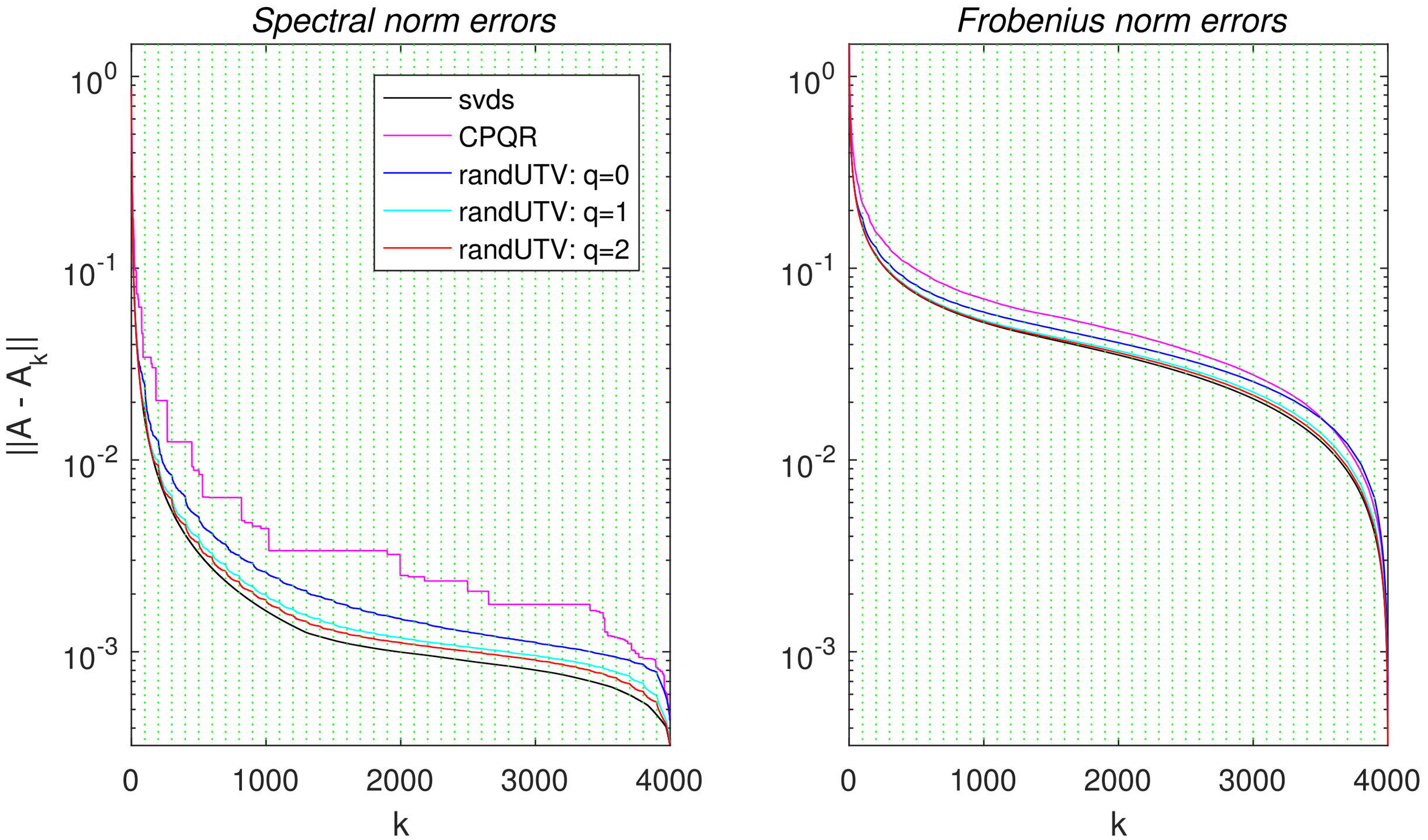




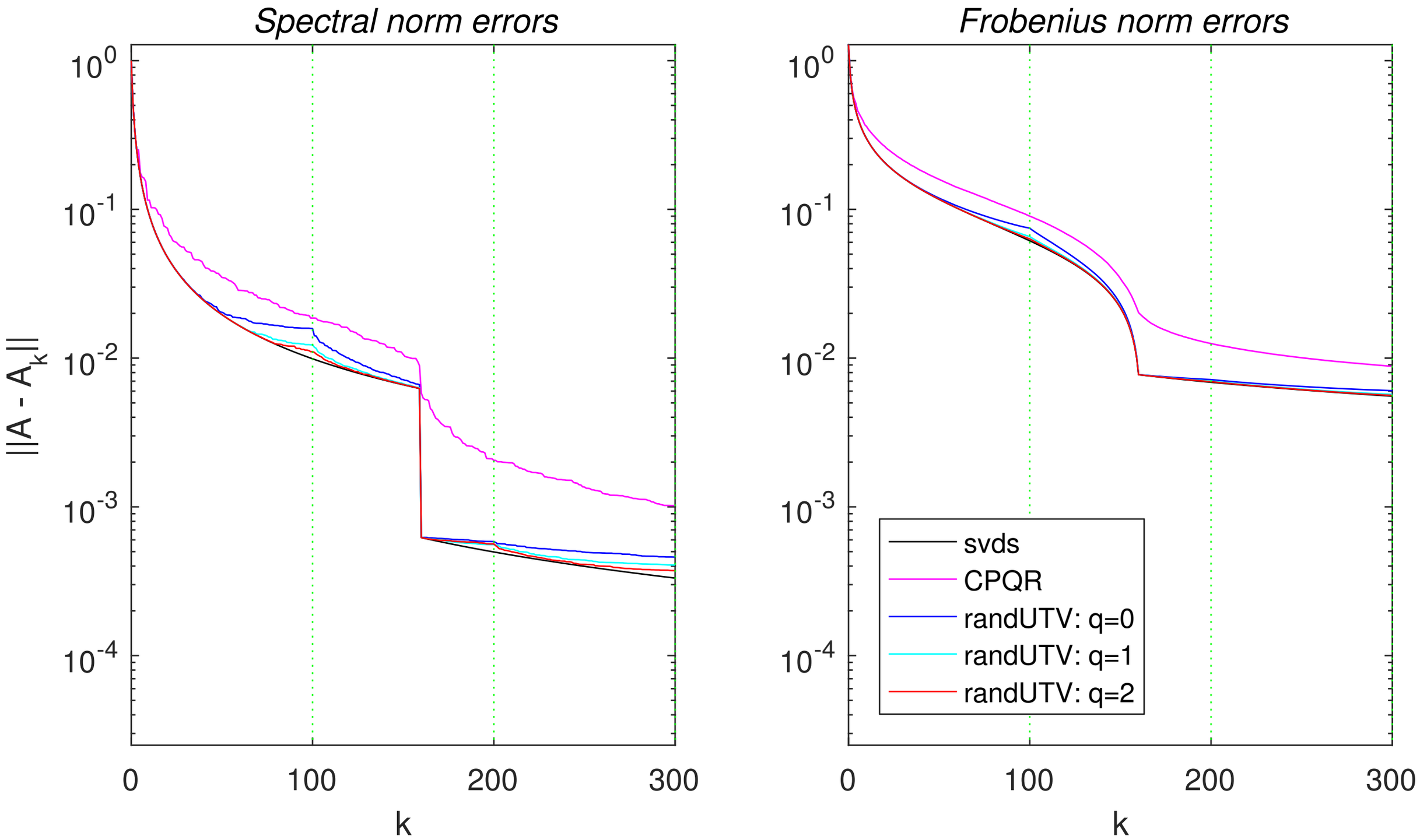
Computational speed of randUTV (red) compared to SVD (black) and column pivoted QR (blue). Each run is for a full factorization of an $n \times n$ matrix. We compare against Intel MKL LAPACK. Observe that the vertical axis is execution time scaled by n^3!



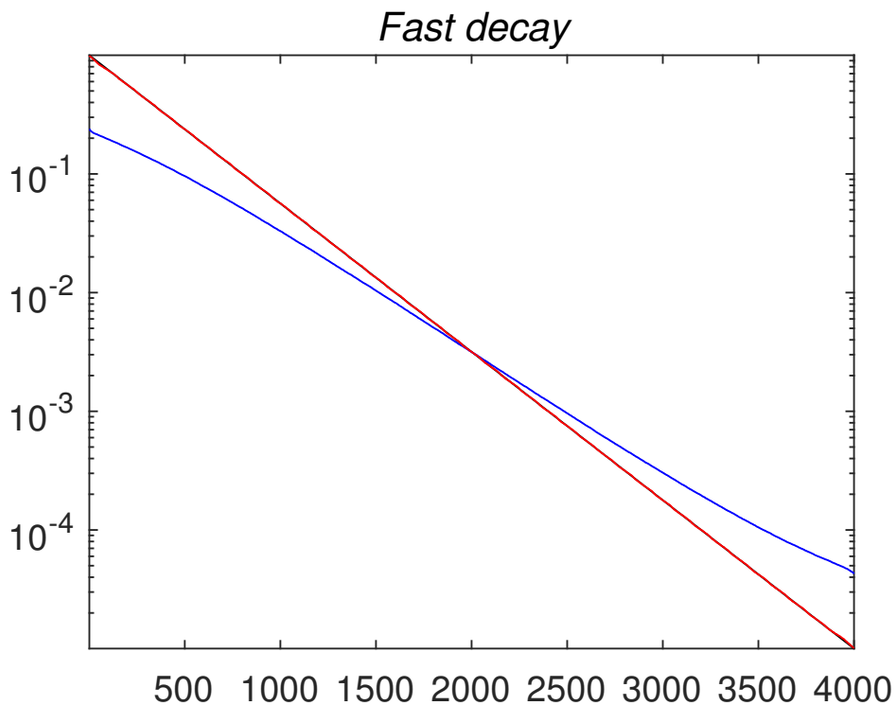
Rank-k approximation errors for the matrix "Fast Decay" of size 4000×4000*. The black lines mark the theoretically minimal errors. The block size was* $b = 100$ *and the green vertical lines mark block limits.*

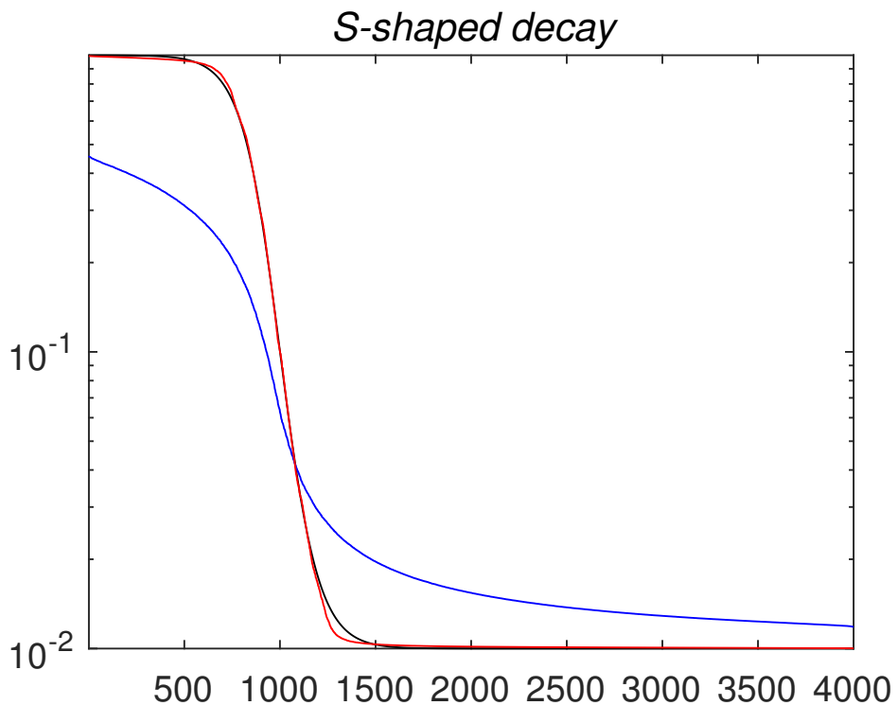


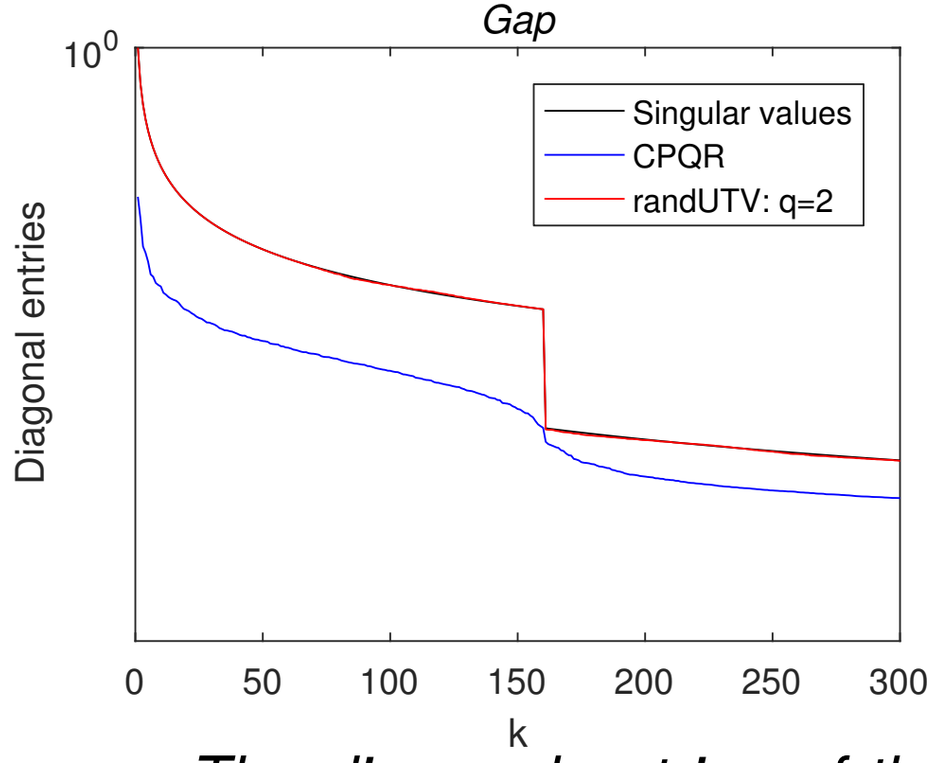
Rank-k approximation errors for the matrix "BIE" of size 4000×4000*. The black lines mark the theoretically minimal errors. The block size was* $b = 100$ *and the green vertical lines mark block limits.*

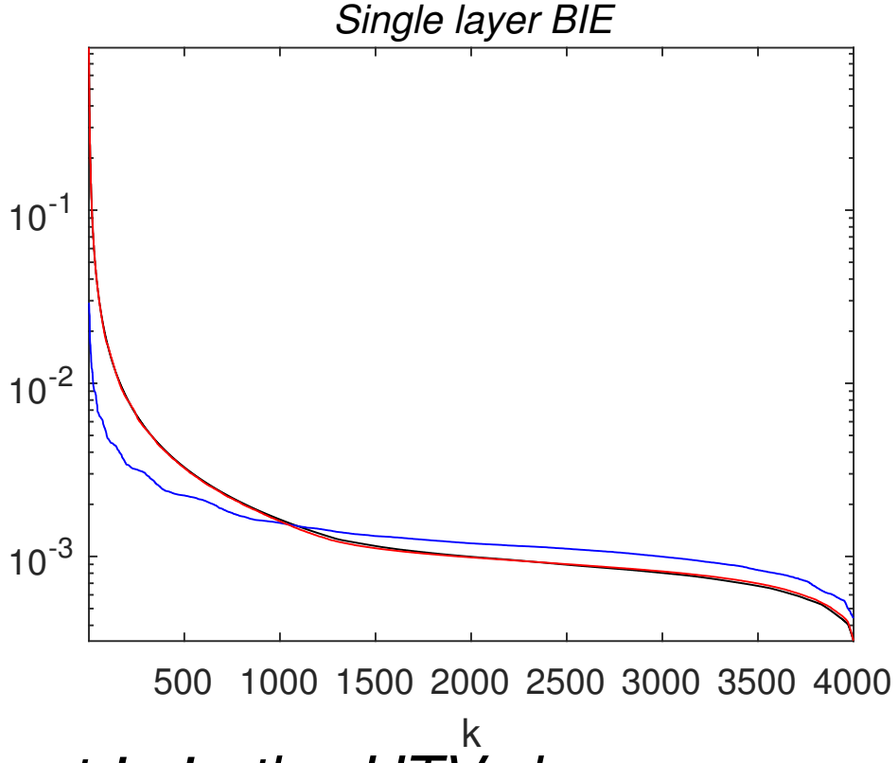


Rank-k approximation errors for $k \leq 300$ *for the matrix "Gap" of size* 4000×4000*. The black lines mark the theoretically minimal errors. The block size was* $b = 100$ *and the green vertical lines mark block limits.*









The diagonal entries of the **T***-matrix in the UTV decomposition (red) provide excellent approximations to the true singular values (black).*

Environment 2: Accelerate FULL factorizations of matrices

Key points:

- All operations are *blocked.*
- Interaction with **A** is only through matrix-matrix multiply.
- Very fast Householder QR with column pivoting. https://github.com/flame/hqrrp/
- Randomized UTV factorization:
  - Accuracy close to SVD.
  - Very fast: similar or faster than CPQR.
  - Admits partial factorizations, given a tolerance.
  - Very communication efficient. On GPU we see $\times 10$ acceleration over SVD.

  To be slightly provocative: *Better than CPQR in basically every respect!*

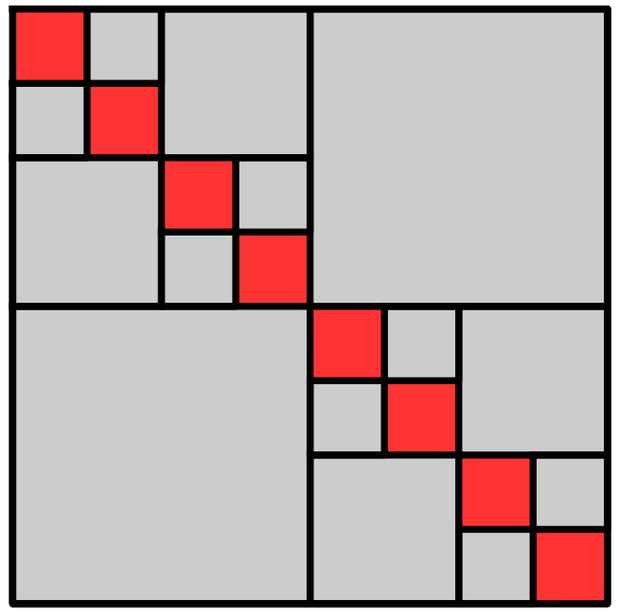
References:

- P.G. Martinsson, *Blocked rank-revealing QR factorizations: How randomized sampling can be used to avoid single-vector pivoting.* arXiv.org report #1505.08115, 2015.
- P.G. Martinsson, Gregorio Quintana-Ortí, Nathan Heavner, and R. van de Geijn, *Householder QR Factorization With Randomization for Column Pivoting (HQRRP).* To appear in SISC.
- P.G. Martinsson, Gregorio Quintana-Ortí, Nathan Heavner, *randUTV: A blocked randomized algorithm for computing a rank-revealing UTV factorization.* Appearing on arXiv within weeks.
- Recent work by Ming Gu and Jed Duersch of UC-Berkeley.

Environment 3: Randomized approximation of rank-structured matrices. (Plug!)

Loosely speaking, a matrix is *rank-structured* if its off-diagonal blocks have low rank to some given precision. These matrices arise upon discretization of integral operators, in accelerating nested dissection, in simulating Monte Carlo processes, etc.

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.



Environment 3: Randomized approximation of rank-structured matrices

Loosely speaking, a matrix is *rank-structured* if its off-diagonal blocks have low rank to some given precision. These matrices arise upon discretization of integral operators, in accelerating nested dissection, in simulating Monte Carlo processes, etc.

Many "formats" have been proposed, including:

- "Fast Multipole Method" matrices.
- $\mathcal{H}$- and $\mathcal{H}^2$-matrices.
- Hierarchically Block Separable (HBS) matrices, a.k.a. "HSS" matrices.
- HODLR matrices (a.k.a. $\mathcal{S}$-matrices).

All these formats allow for (more or less) efficient matrix computations involving a range of operations such as matrix-vector multiply, matrix-matrix multiply, LU factorization, matrix inversion, forming of Schur complements, etc.

Objective: Suppose a matrix $\mathbf{A}$ is rank-structured, that you are *given* a tessellation pattern, and that you have an efficient technique for evaluating the matrix-vector product $\mathbf{x} \mapsto \mathbf{Ax}$. We then seek to build all factors in the rank-structured representation of $\mathbf{A}$.

Applications: Build "frontal matrices" in nested dissection. Matrix-matrix multiplication of two structured matrices. Convert from, say, FMM format, to HBS format. Et cetera.

Let $\mathbf{A}$ be a rank-structured matrix, for which we can rapidly evaluate $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ and $\mathbf{x} \mapsto \mathbf{A}^*\mathbf{x}$.

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

- P.G. Martinsson, *A fast randomized algorithm for computing a Hierarchically Semi-Separable representation of a matrix.* 2008 arxiv report. 2011 SIMAX paper.
- Later improvements by Jianlin Xia, Sherry Li, etc.

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

The method is still very fast in many situations, and can save messy coding work. For instance, implementing the matrix-matrix mutliplication, or changing the partition tree, are quite hard to implement efficiently.

- L. Lin, J. Lu, L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, JCP, **230**(10), 2011.
- P.G. Martinsson, *Compressing rank-structured matrices via randomized sampling.* SISC, **38**(4), 2016.

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 $\mathbf{R}_2 = \mathbf{V}_2^*\,\mathbf{R}$.

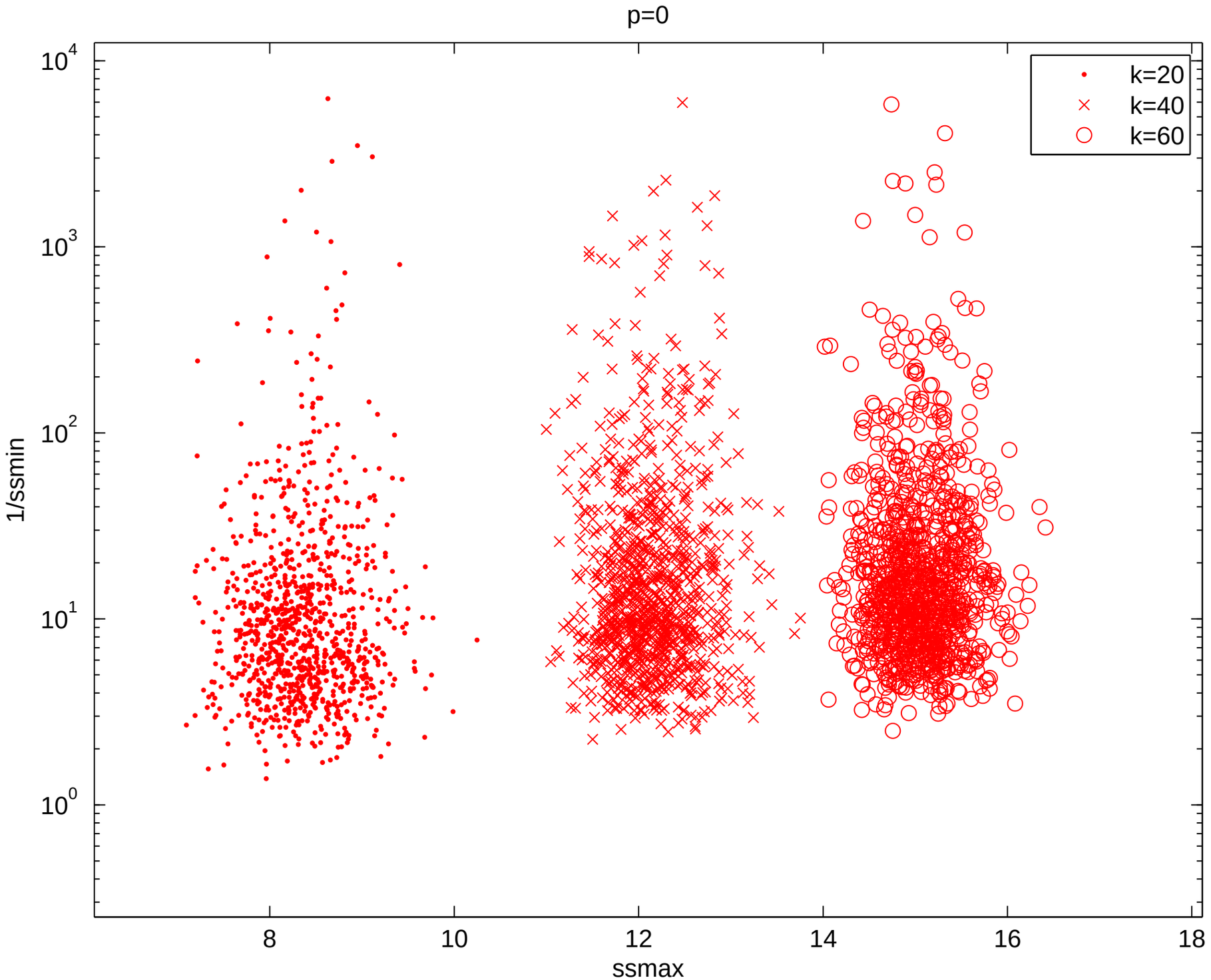
Assumption: **R** is drawn from a normal Gaussian distribution.

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?

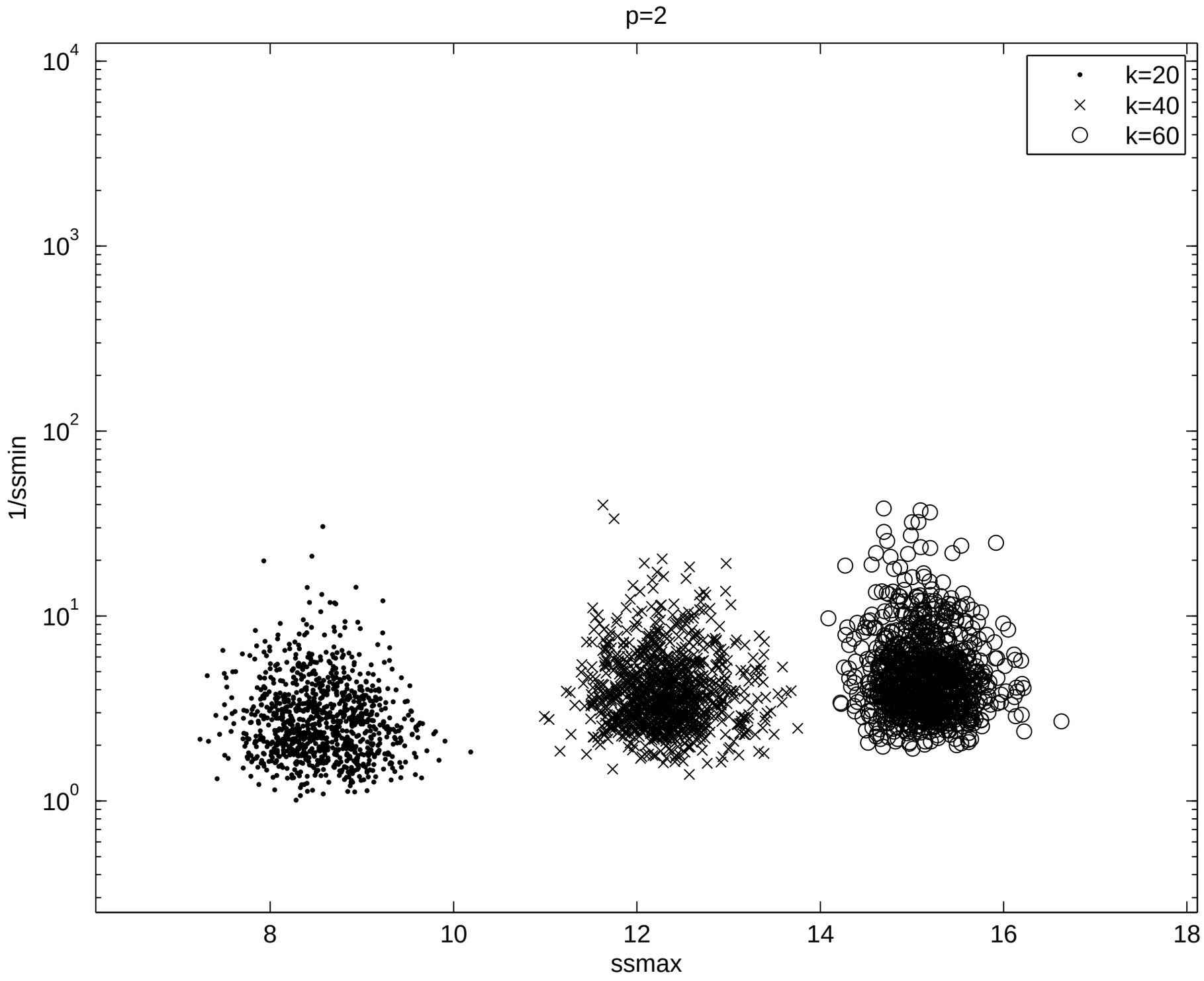
If $p = 0$, then $\|\mathbf{R}_1^\dagger\|$ is typically large, and is very unstable.

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



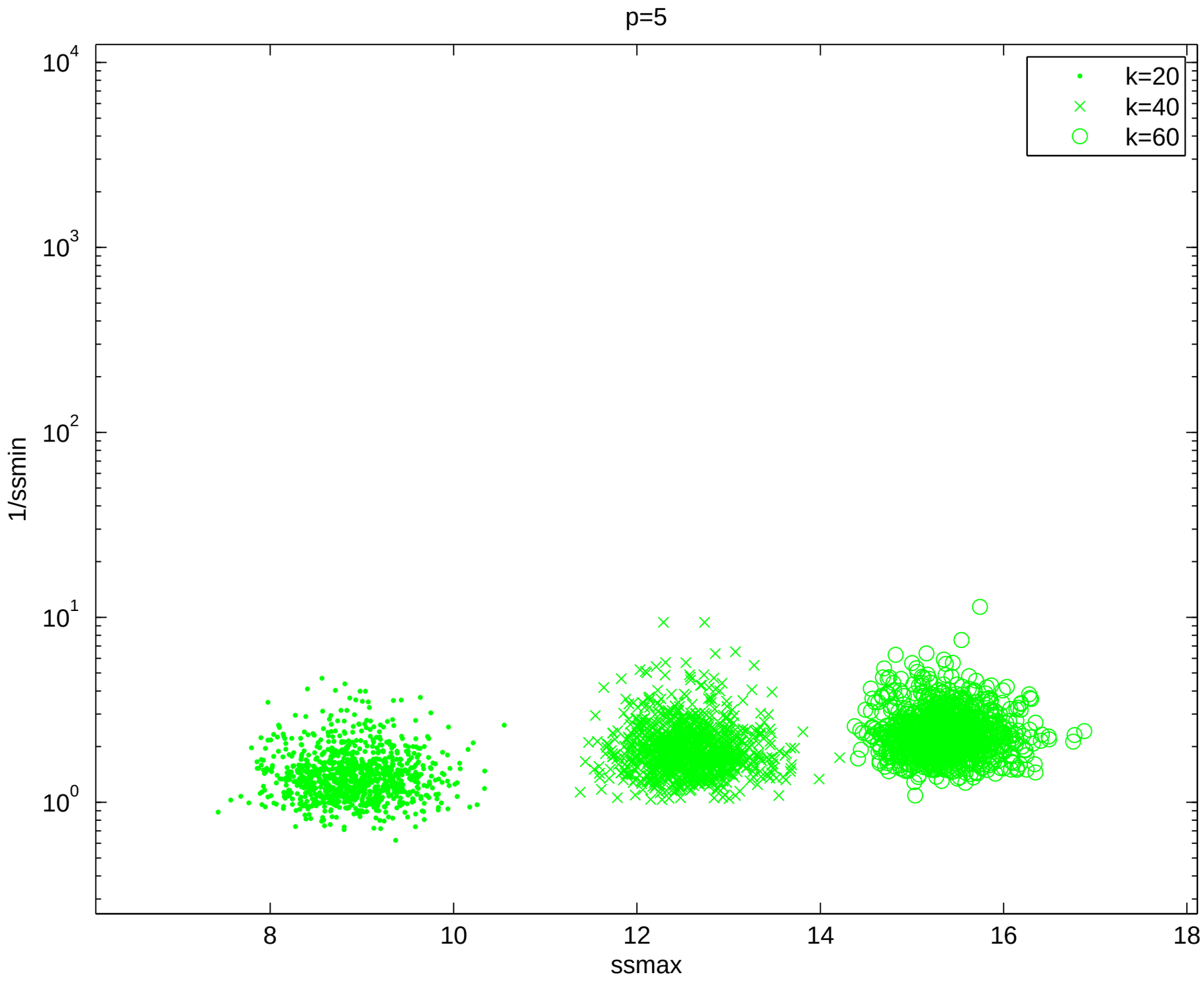
$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

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



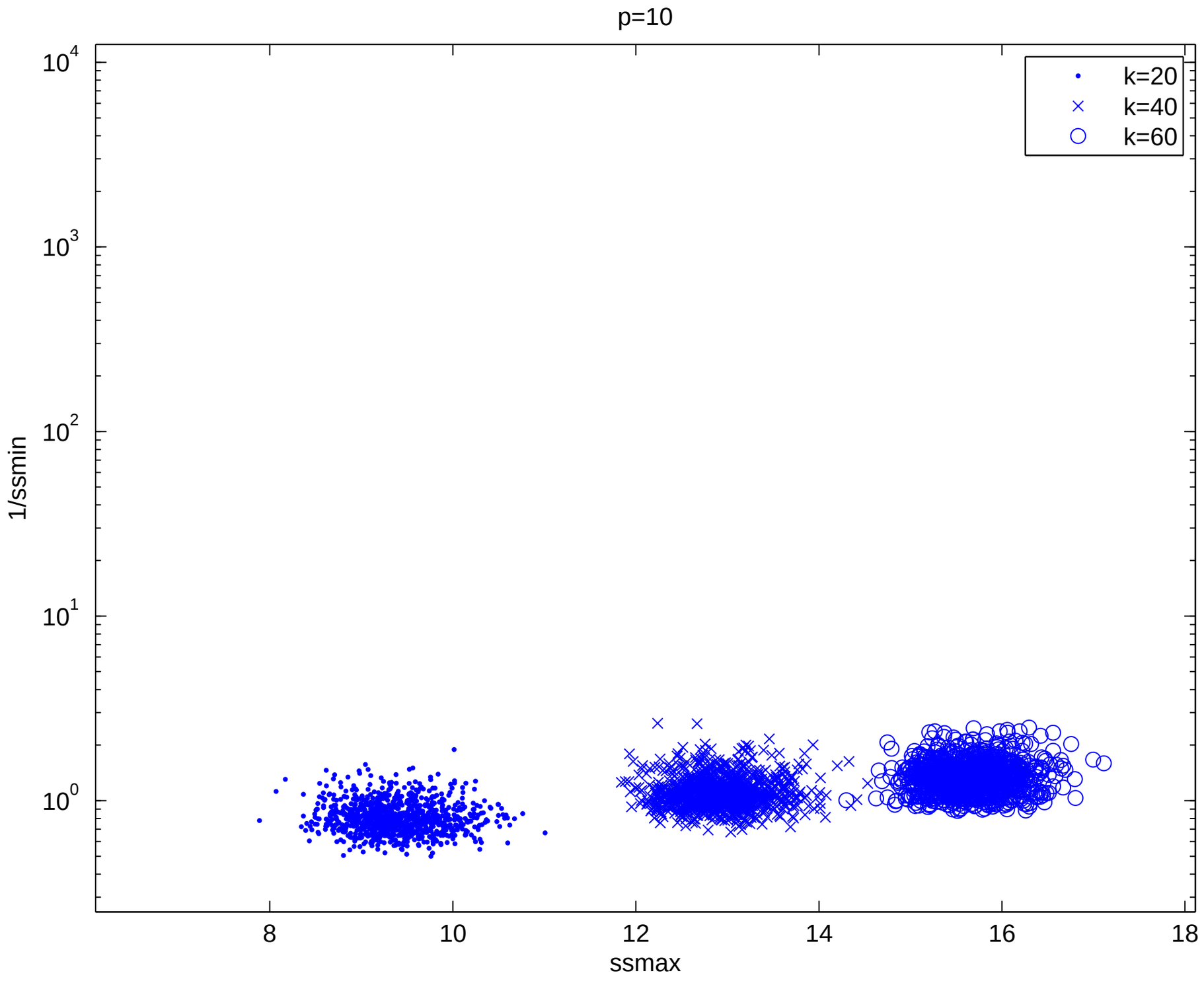
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

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



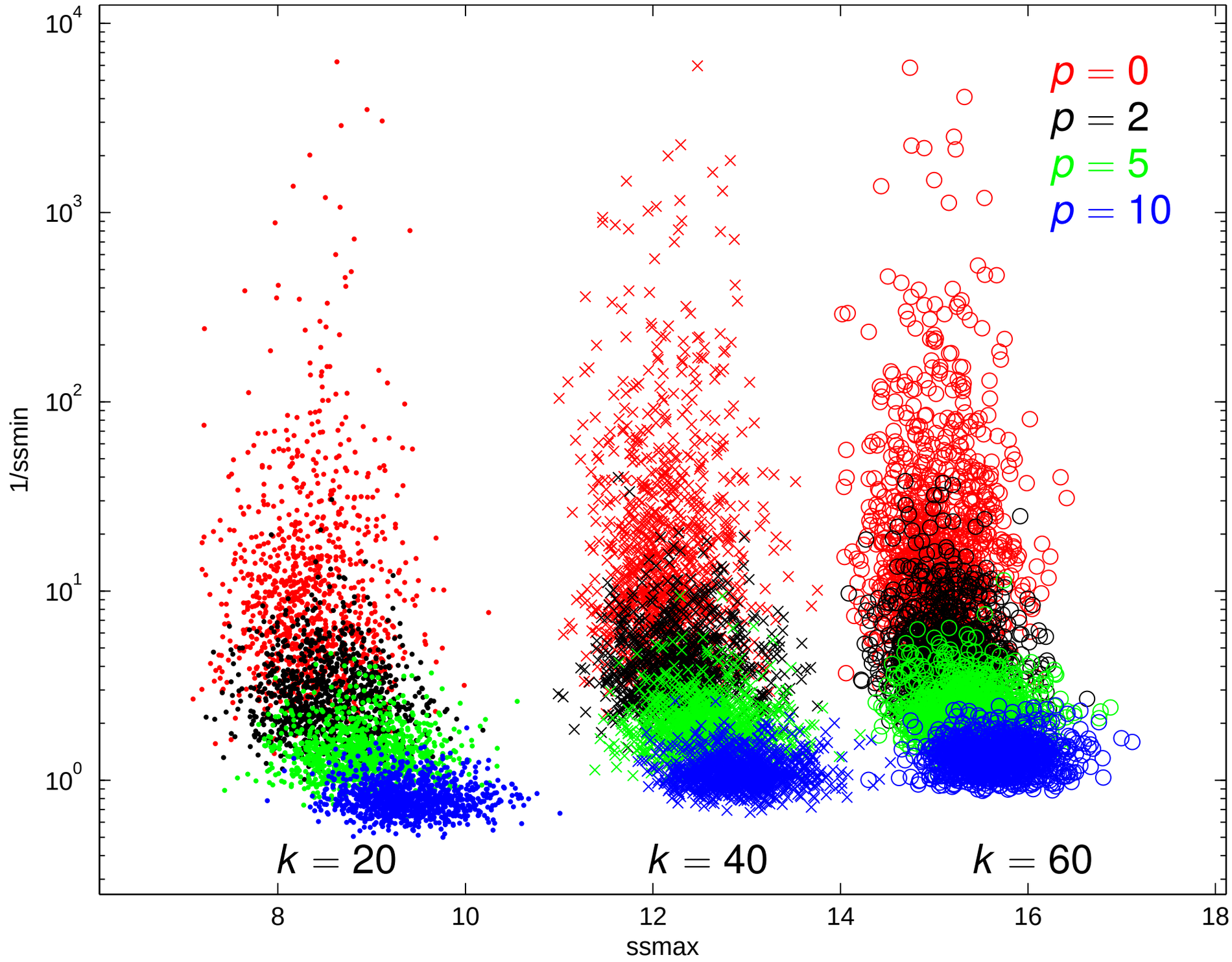
$1/\sigma_{\min}$ is plotted against $\sigma_{\max}$.

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)

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

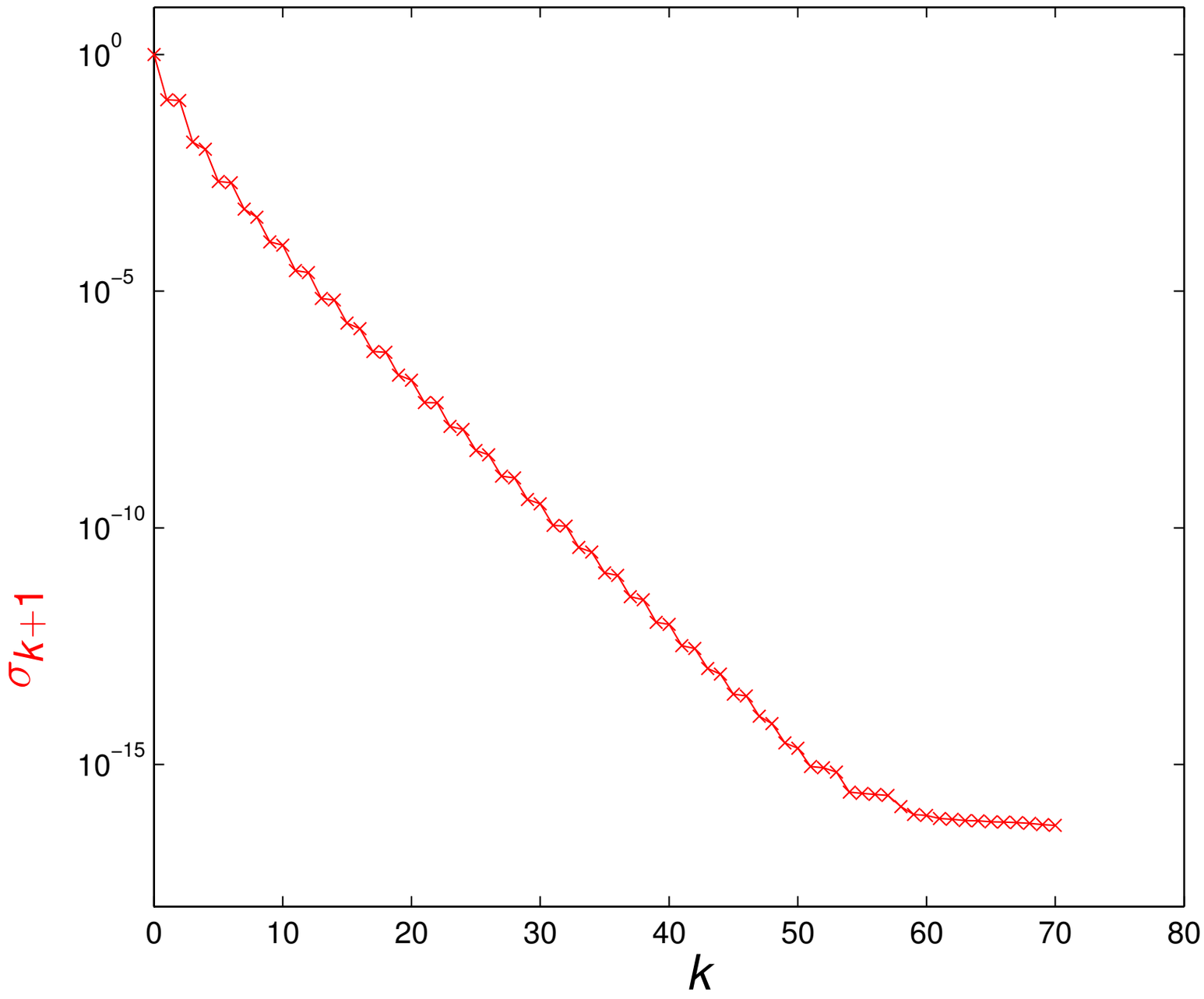
When E_t holds true, we have bounds on the "badness" of $\mathbf{R}_1^\dagger$:

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

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:



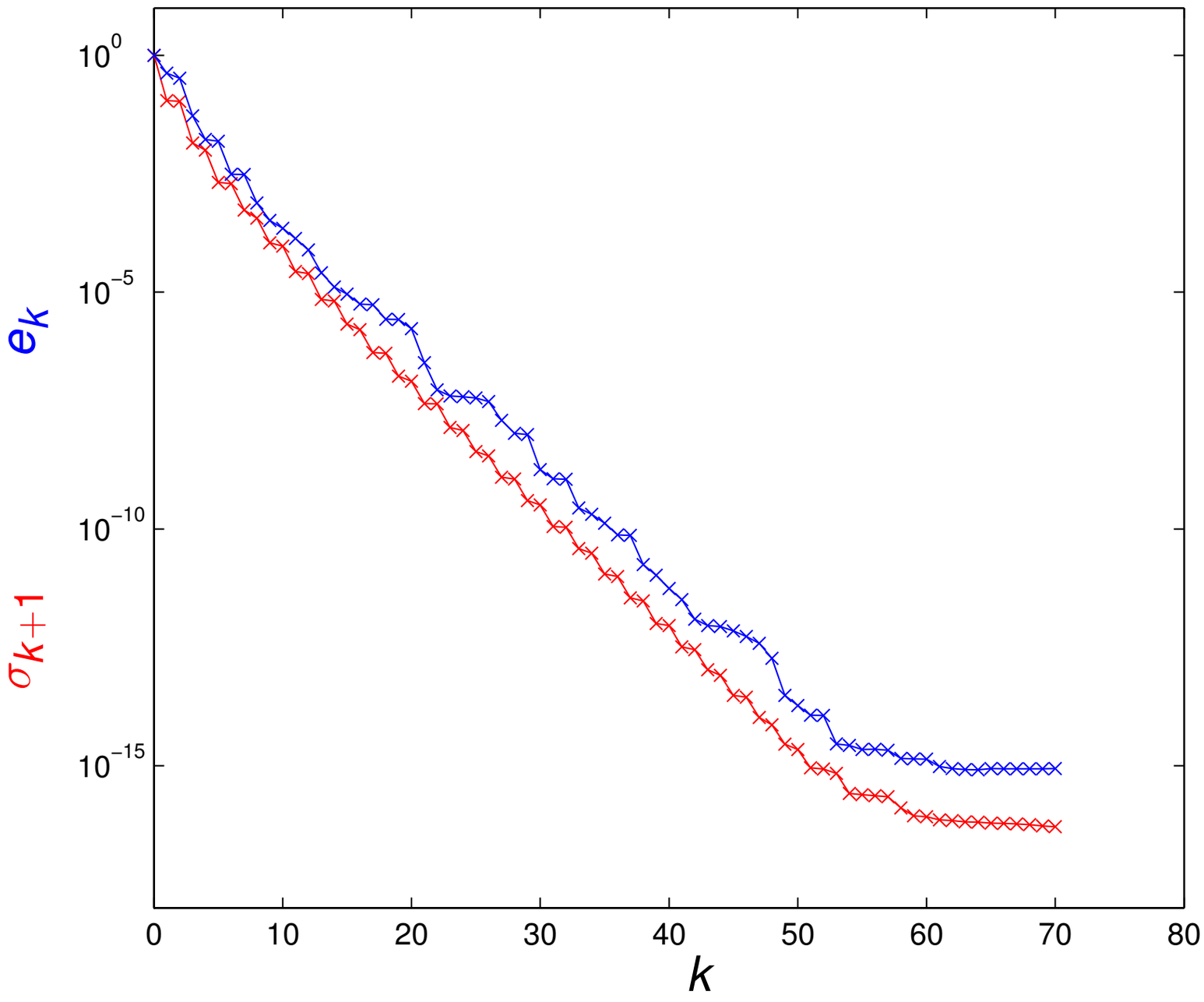
The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

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.

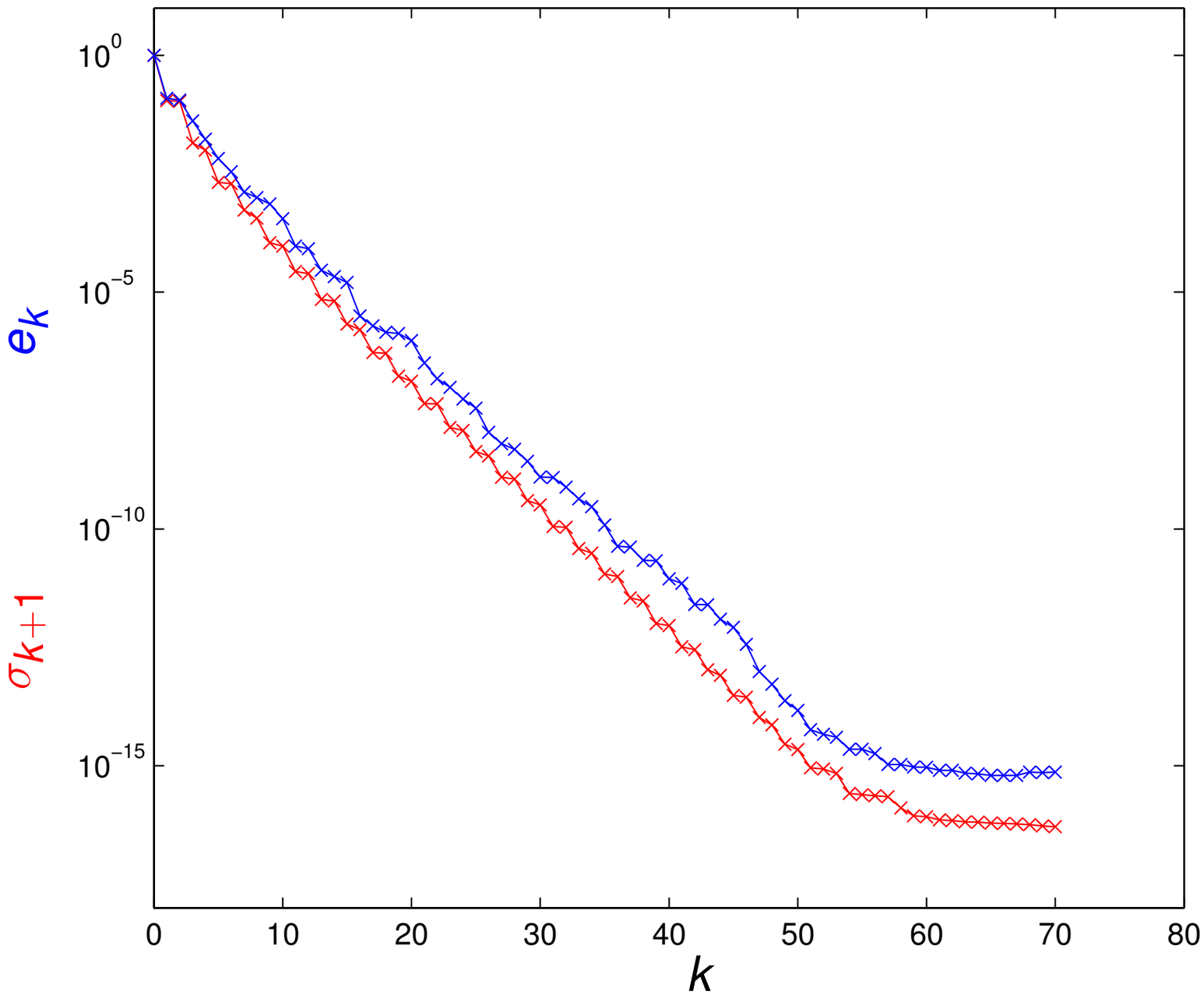
The blue line indicates the actual errors e_k incurred by one instantiation of the proposed method.

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.

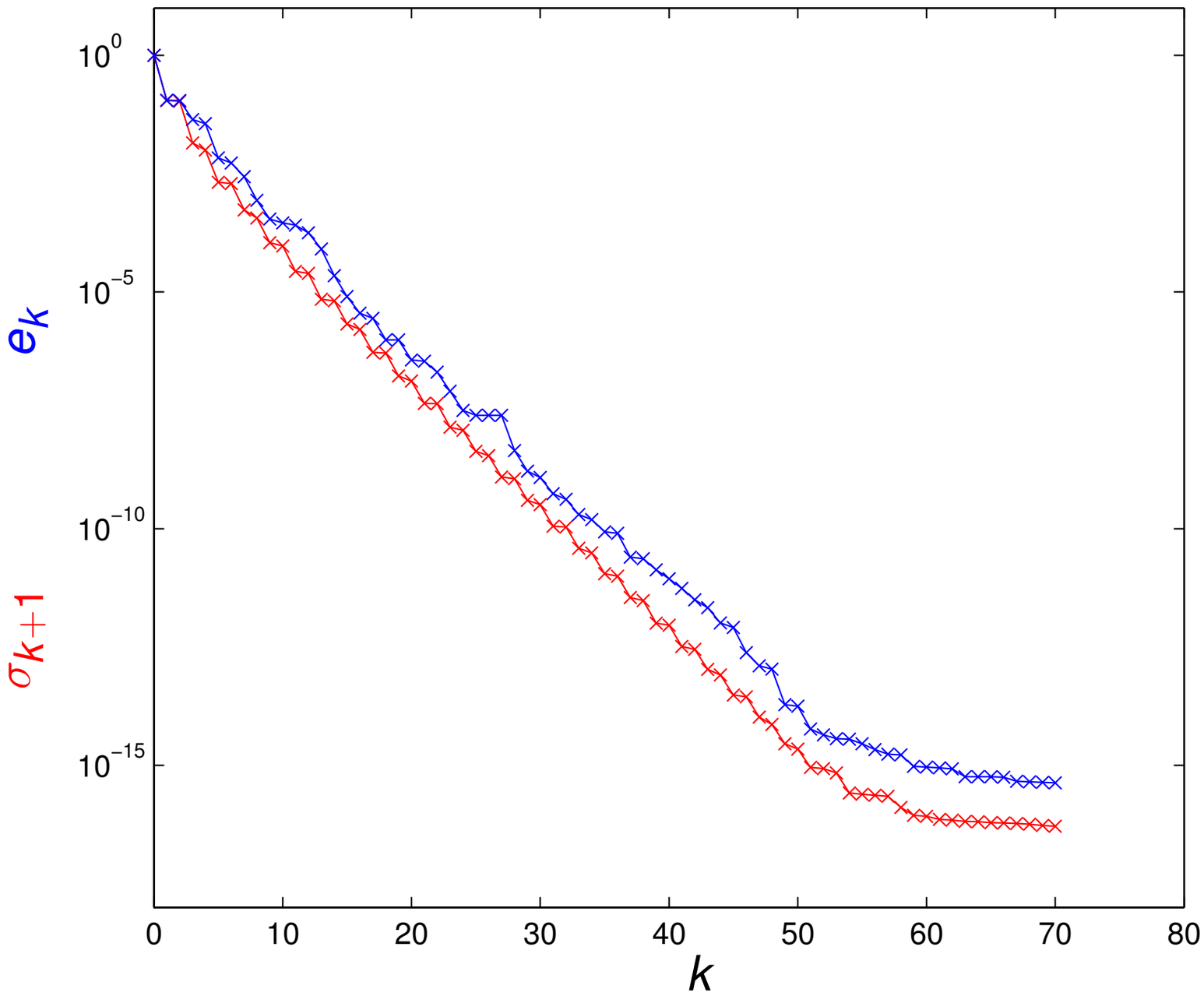
The blue line indicates the actual errors e_k incurred by a different instantiation of the proposed method.

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.

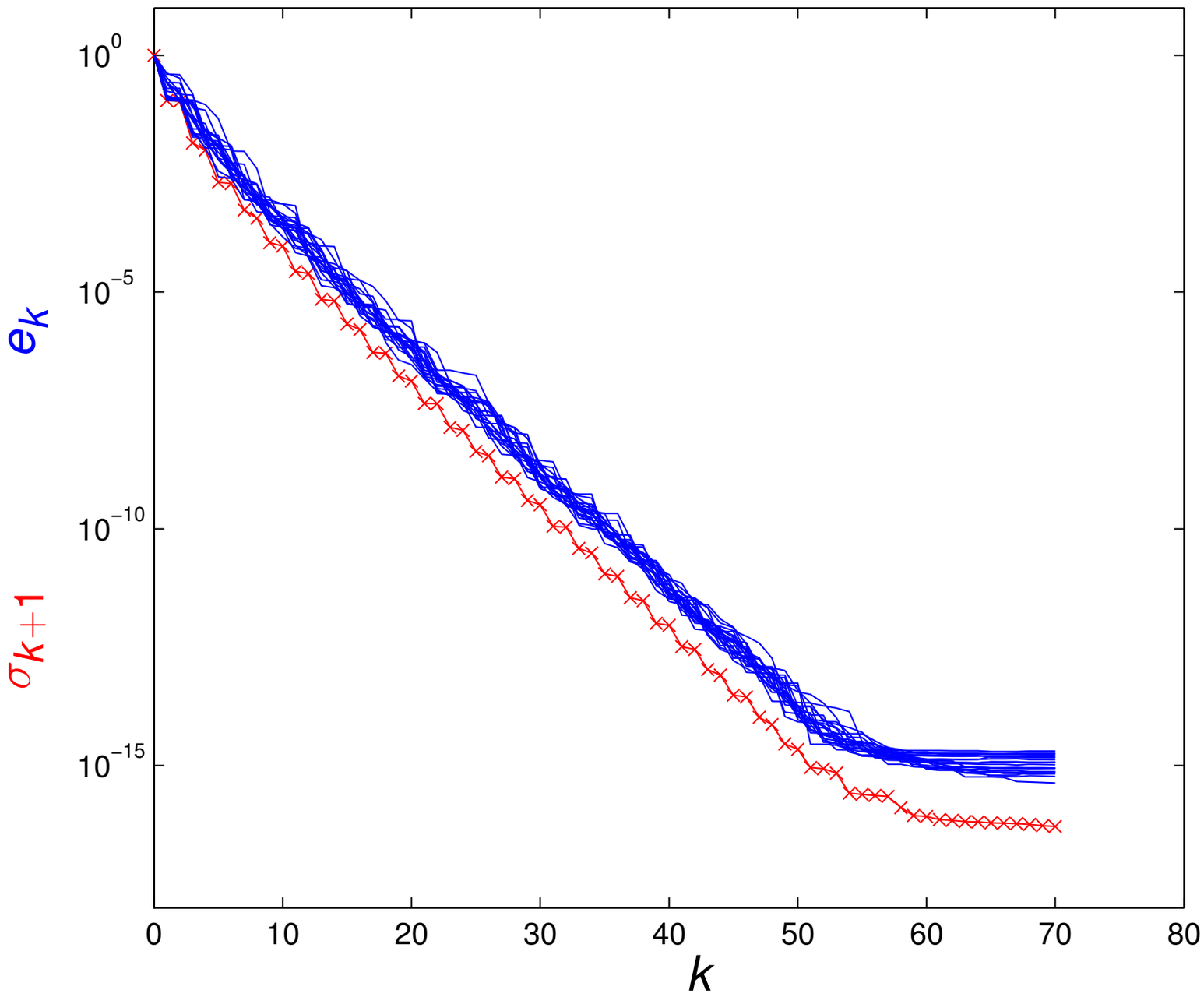
The blue line indicates the actual errors e_k incurred by a different instantiation of the proposed method.

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.

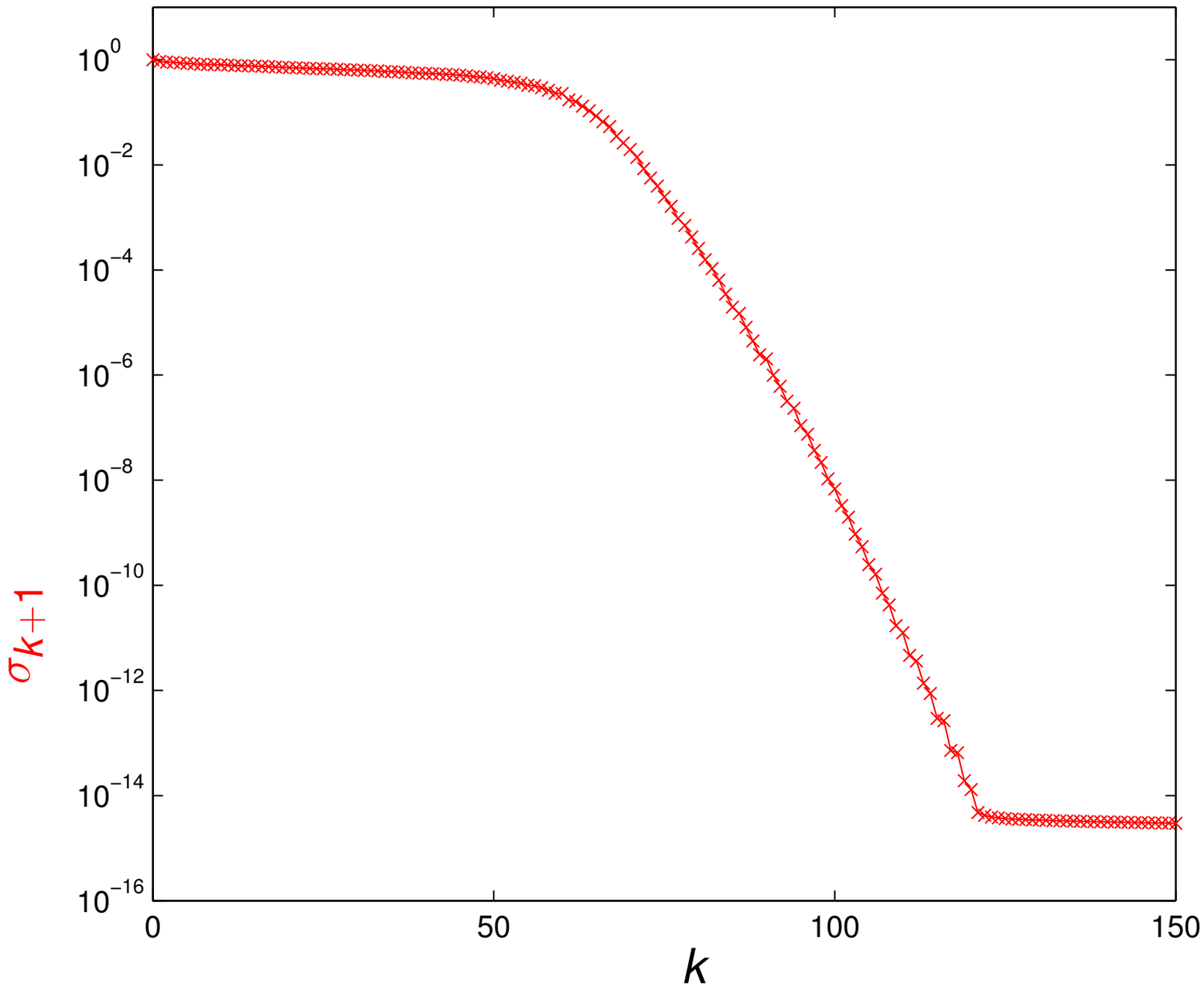
The blue lines indicate the actual errors e_k incurred by 20 instantiations of the proposed method.

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:



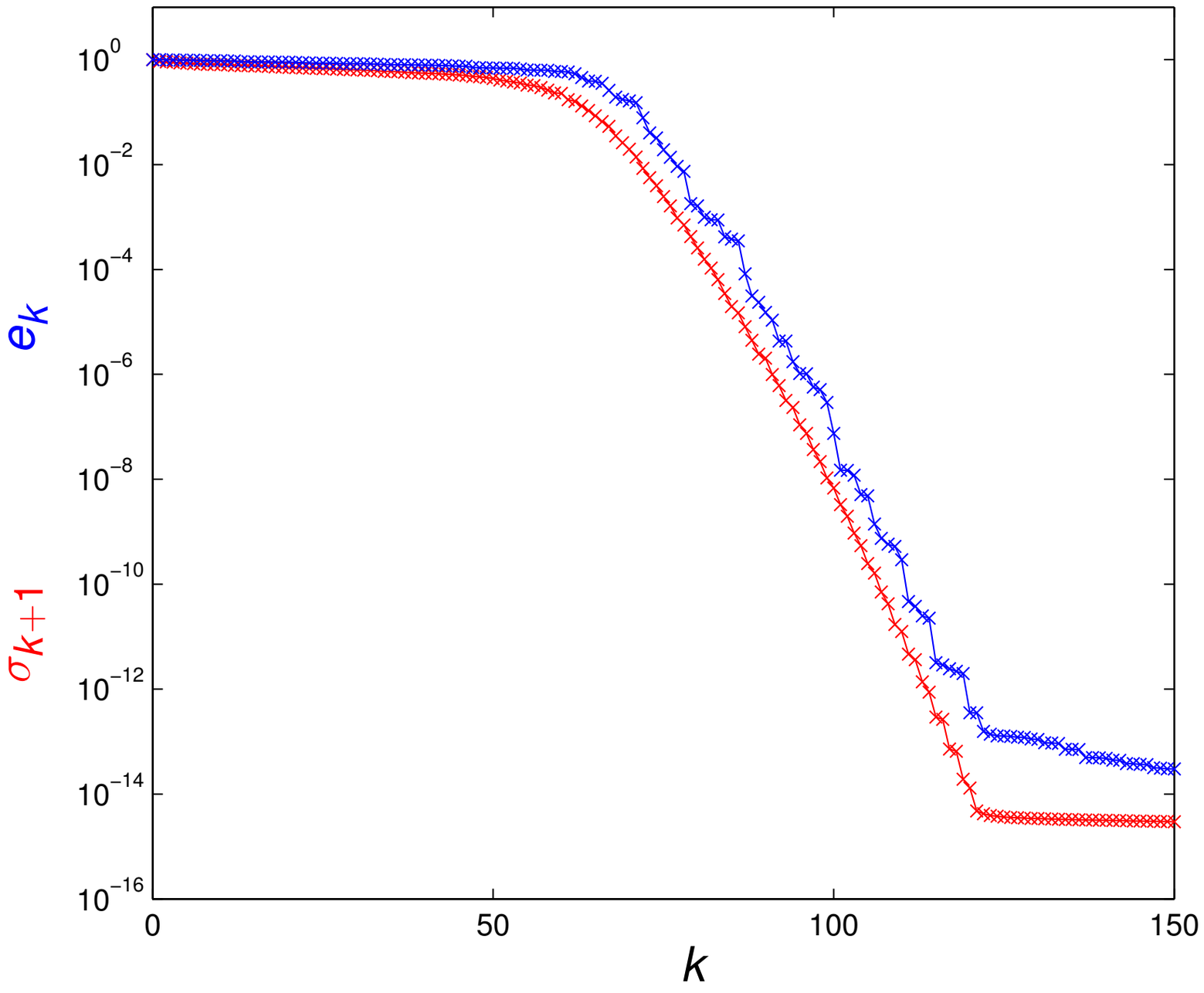
The red line indicates the singular values σ_{k+1} of **A**. These indicate the theoretically minimal approximation error.

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.

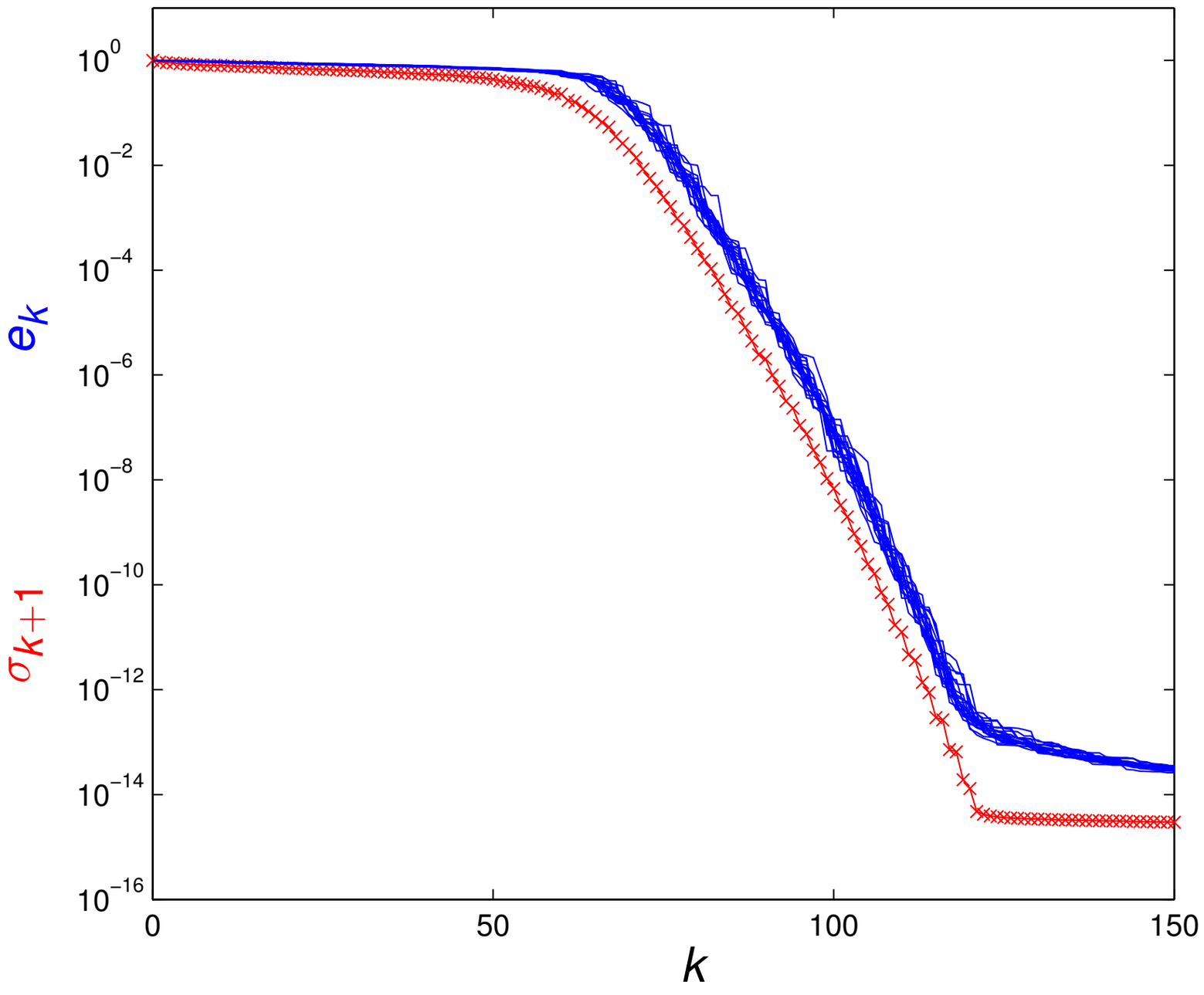
The blue line indicates the actual errors e_k incurred by one instantiation of the proposed method.

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.

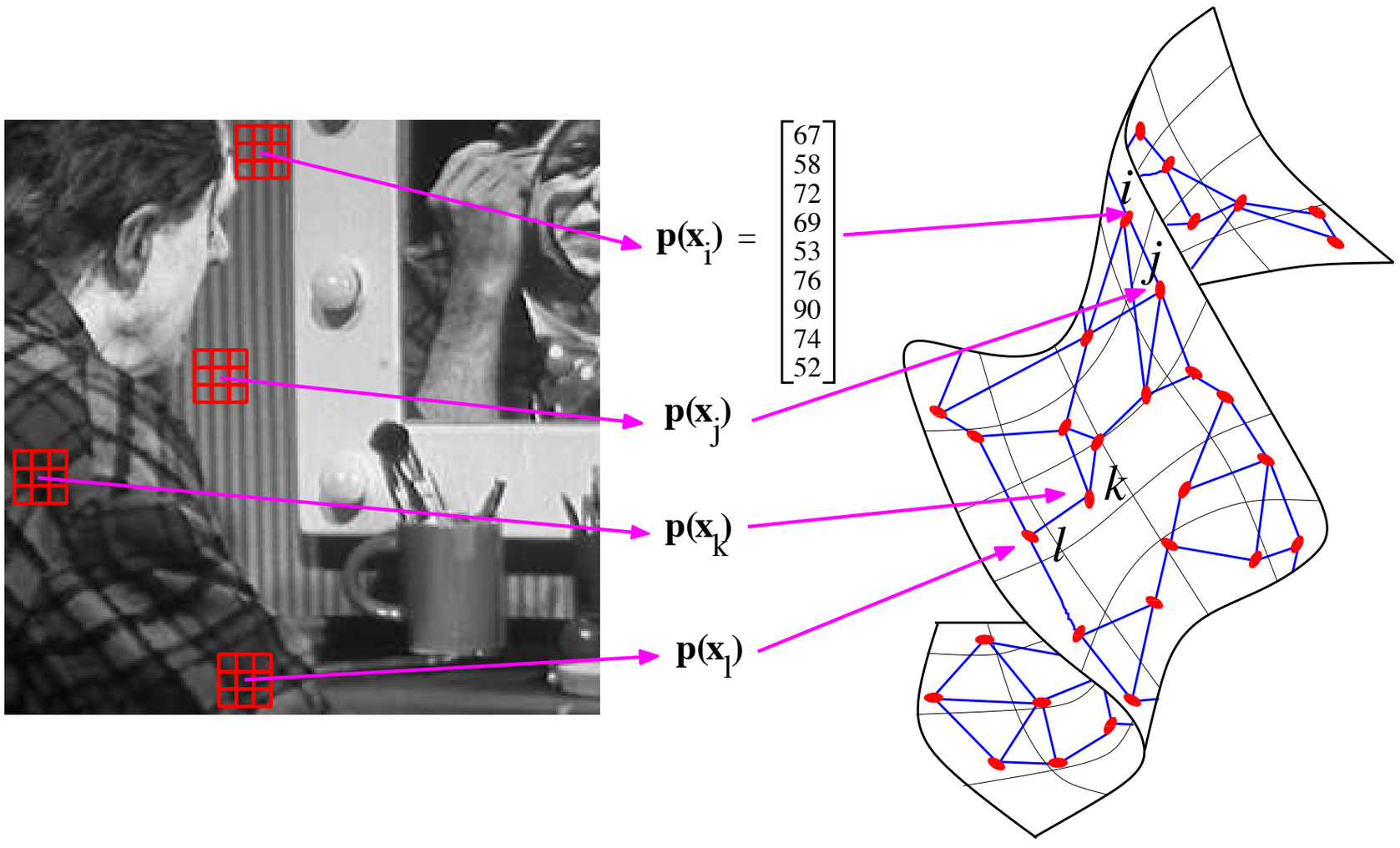
A is a discrete approximation of a certain compact integral operator normalized so that $\|\mathbf{A}\| = 1$.

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

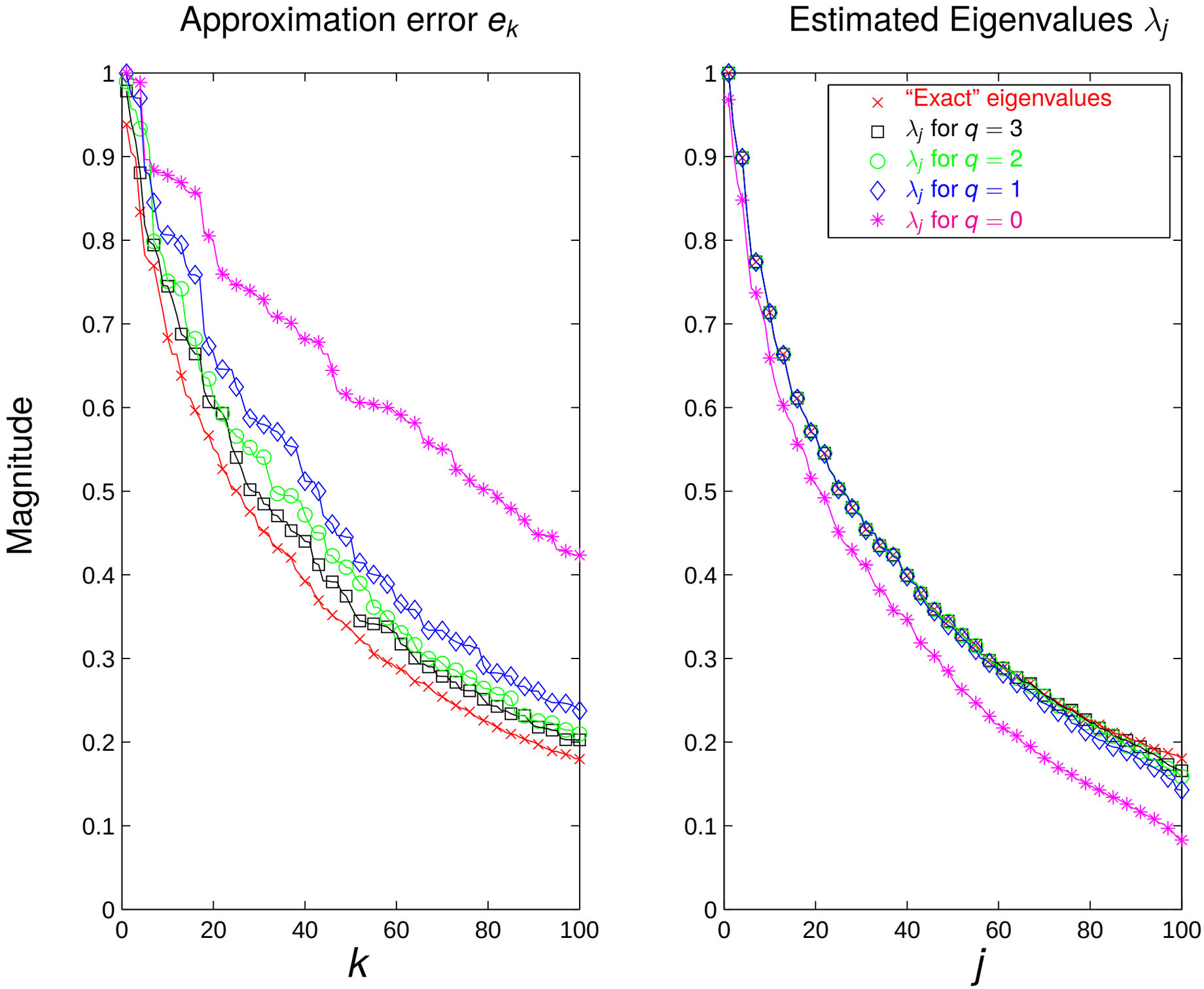
Example 3:

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.

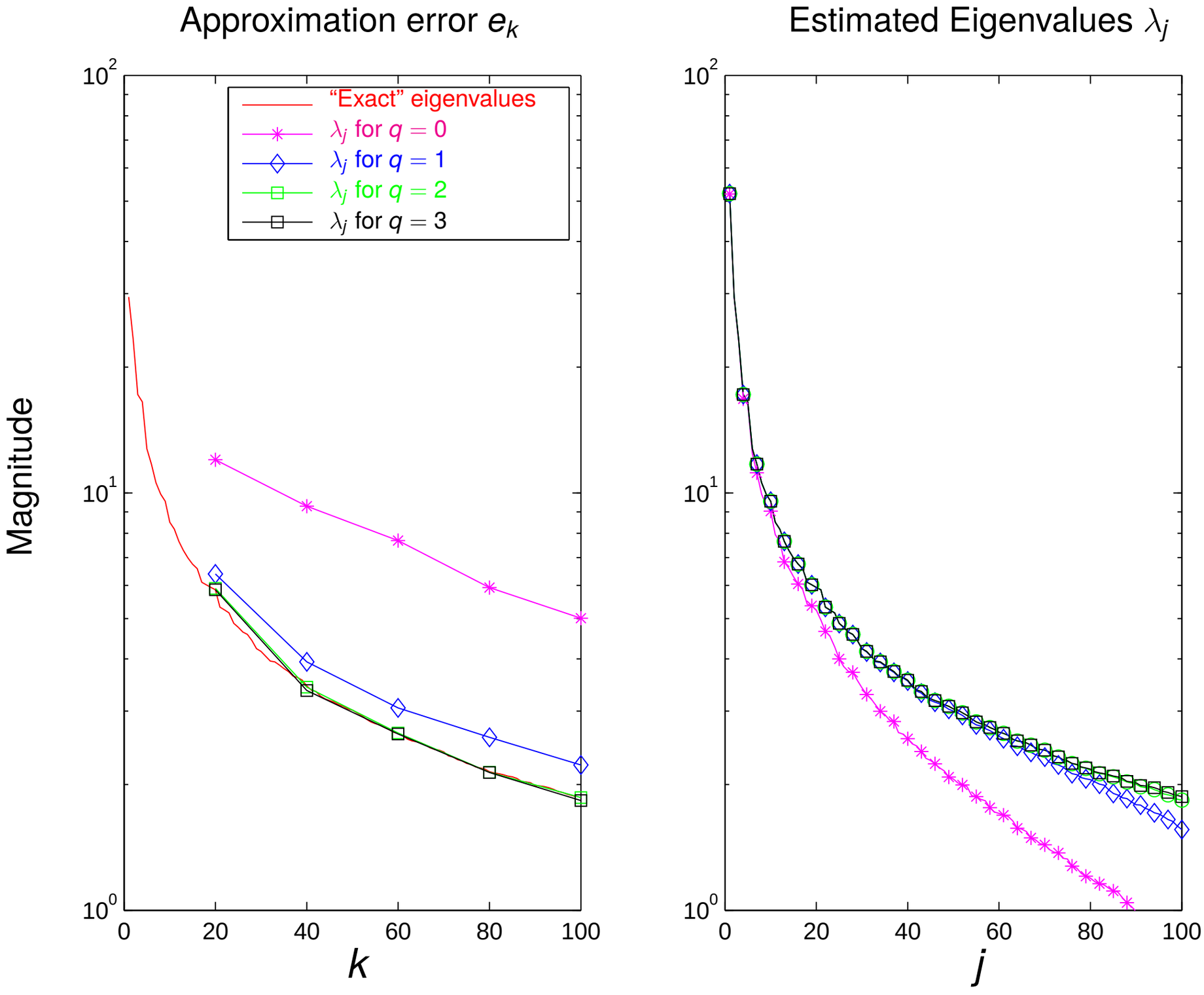
Example 4: “Eigenfaces”

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.

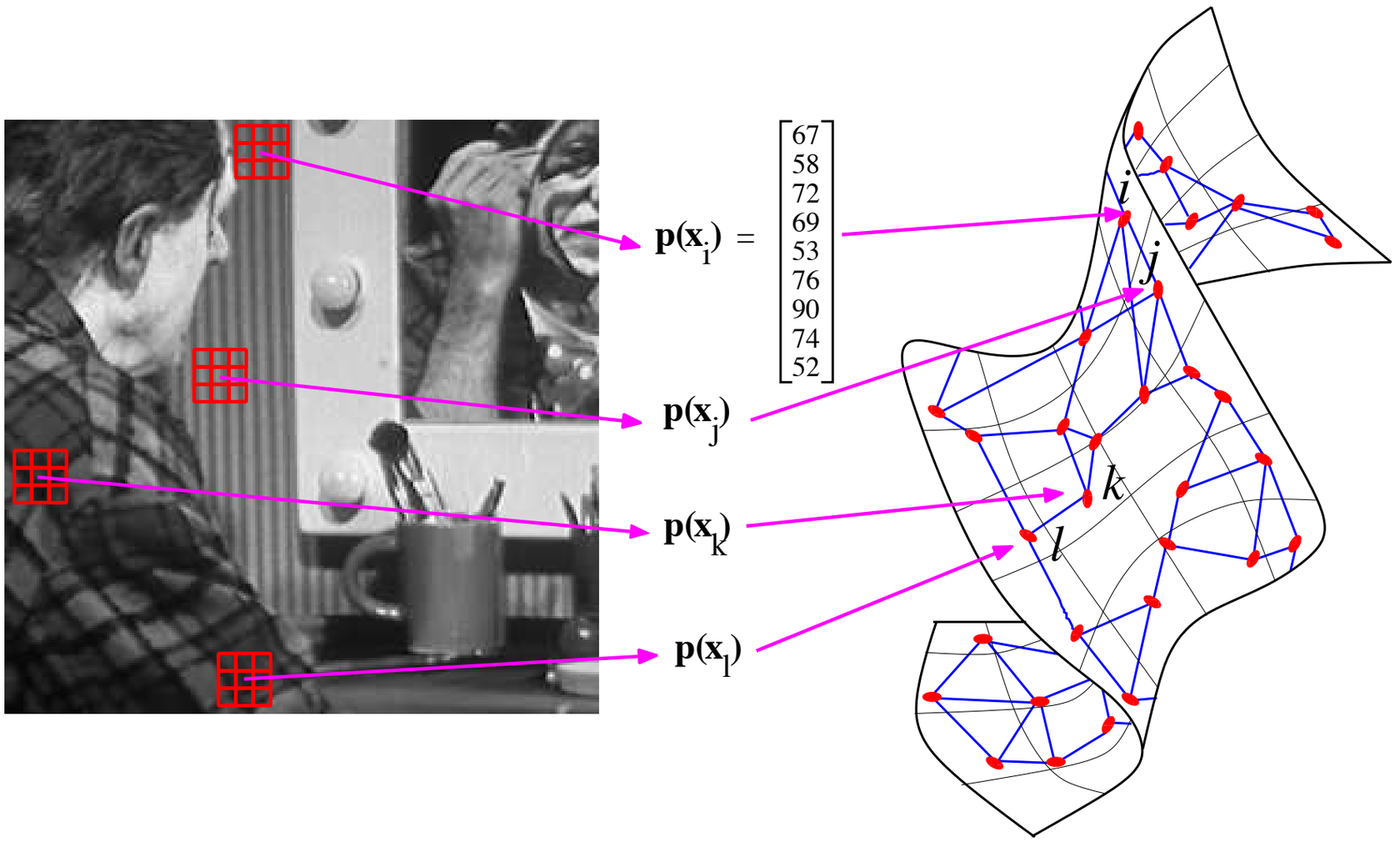
The novelty is the error analysis, and the finding that using a very small q is often fine.

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

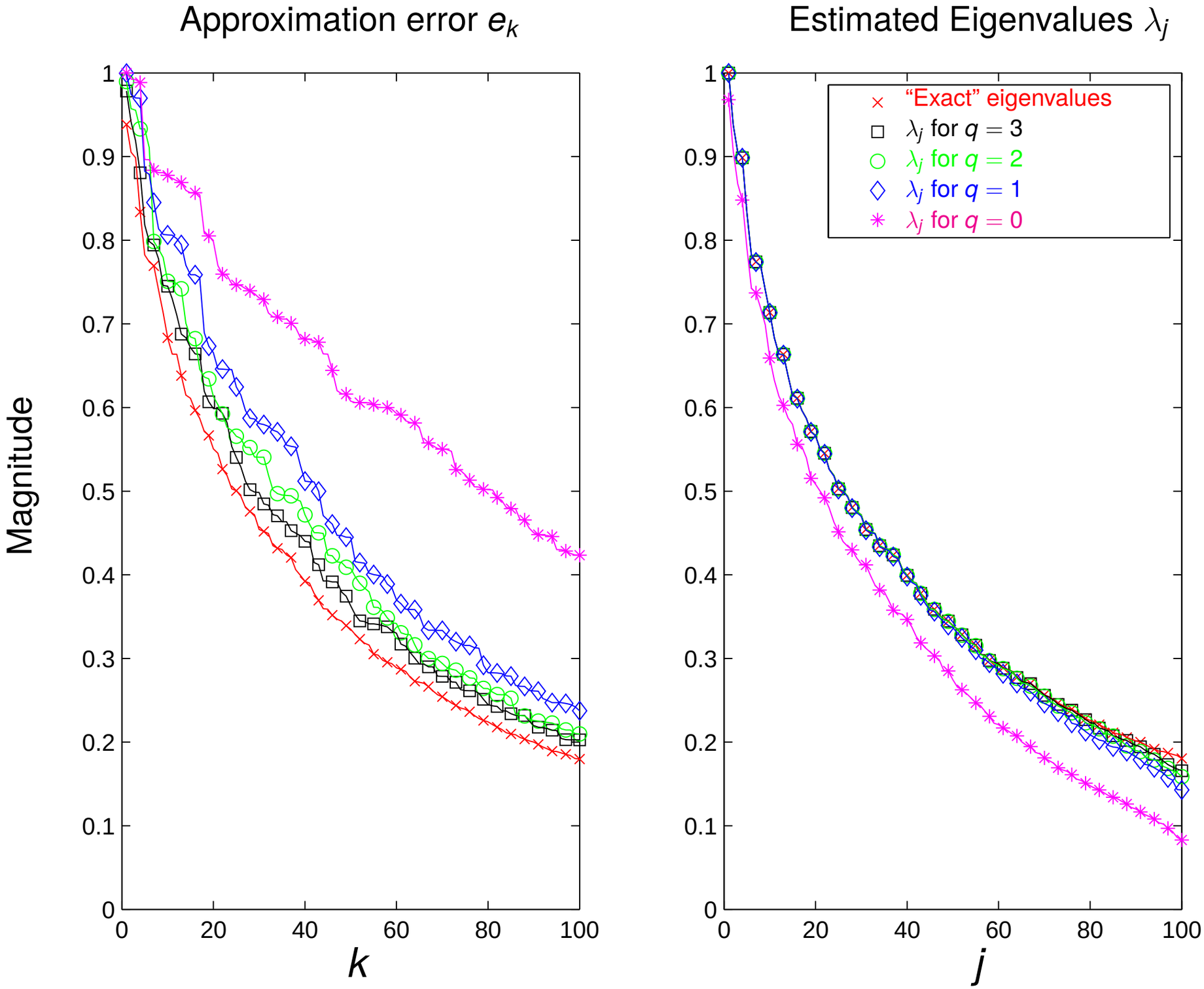
Example 3 (revisited):

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!

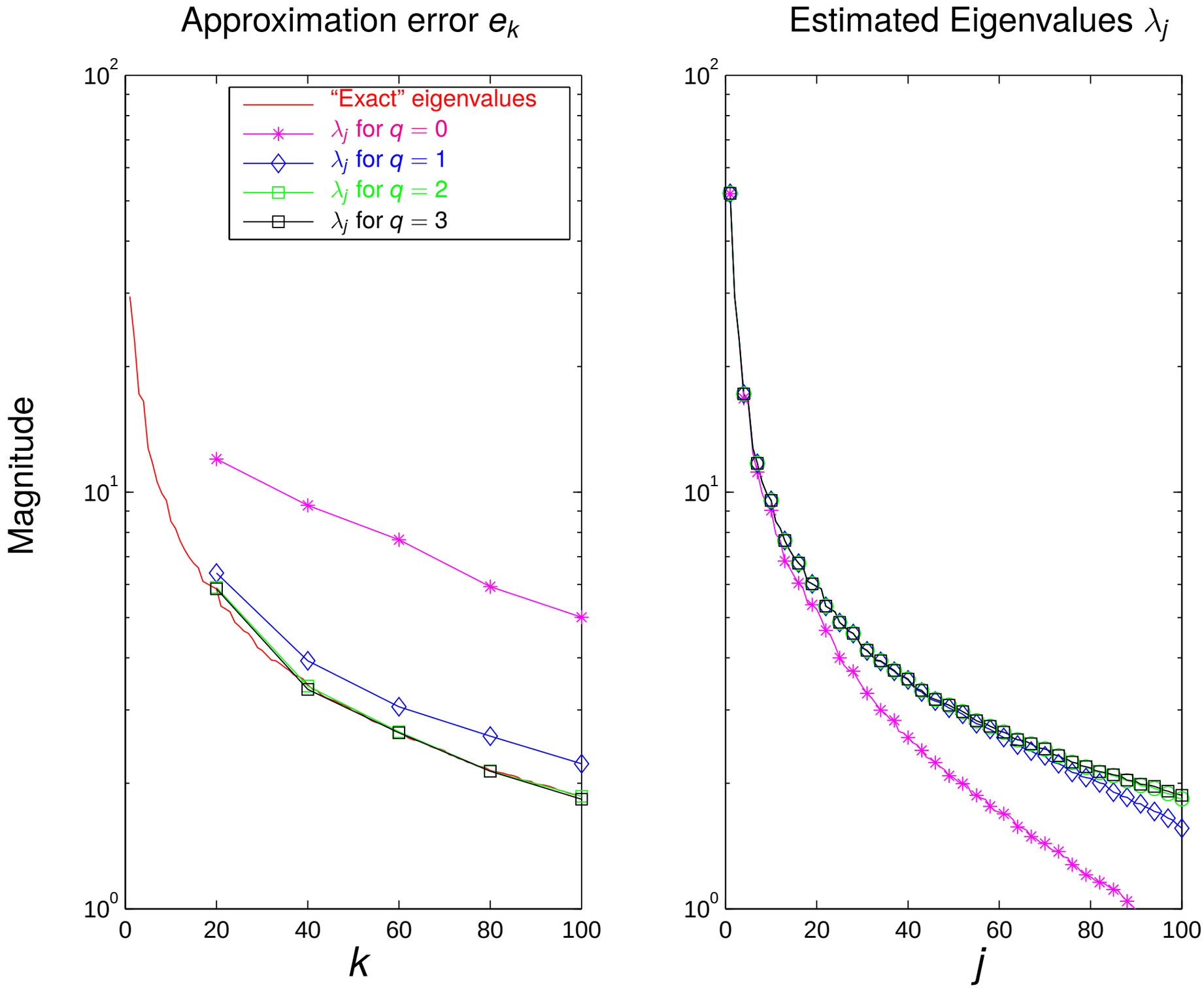
Example 4 (revisited): “Eigenfaces”

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:

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. *[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.)

4. *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.

Current work:

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. *[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.)

4. *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.


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". 2006 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.