Fast matrix computations via randomized sampling

Per-Gunnar Martinsson
The University of Colorado at Boulder

Ph.D. Students:

Adrianna Gillman

Nathan Halko

Patrick Young

Collaborators:

Edo Liberty (Yale)

Vladimir Rokhlin (Yale)

Yoel Shkolnisky (Yale)

Joel Tropp (Caltech)

Mark Tygert (UCLA/Courant)

Franco Woolfe (Goldman Sachs)

Notation:

A vector $x \in \mathbb{R}^n$ is measured using the ℓ^2 (Euclidean) norm:

$$||x|| = \left(\sum_{j=1}^{n} x_j^2 \right)^{1/2}.$$

A matrix $A \in \mathbb{R}^{m \times n}$ is measured using the corresponding operator norm:

$$||A|| = \sup_{x \neq 0} \frac{||A\,x||}{||x||}.$$

Low-rank approximation

An $N \times N$ matrix A has rank k if there exist matrices B and C such that

$$\begin{array}{ccccc} A & = & B & & C. \\ N \times N & & N \times k & & k \times N \end{array}$$

When $k \ll N$, computing the factors B and C is advantageous:

- Storing B and C require $O(N\,k)$ storage instead of $O(N^2)$.
- A matrix-vector multiply requires $2\,N\,k$ flops instead of N^2 flops.
- Certain factorizations reveal properties of the matrix.

In actual applications, we are typically faced with approximation problems:

Problem 1: Given a matrix A and a precision ε, find the minimal k such that

$$\min\{||A - \tilde{A}|| : \text{ rank}(\tilde{A}) = k\} \leq \varepsilon.$$

Problem 2: Given a matrix A and an integer k, determine

$$A_k = \text{argmin}\{||A - \tilde{A}|| : \text{ rank}(\tilde{A}) = k\}.$$

The singular value decomposition (SVD) provides the exact answer.

Any $m \times n$ matrix A admits a factorization (assuming $m \geq n$)

$$A = U\,D\,V^{\mathrm{t}} = [u_1\ u_2\ \cdots\ u_n] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^{\mathrm{t}} \\ v_2^{\mathrm{t}} \\ \vdots \\ v_n^{\mathrm{t}} \end{bmatrix} = \sum_{j=1}^{n} \sigma_j\, u_j\, v_j^{\mathrm{t}}.$$

σ_j is the j'th "singular value" of A
u_j is the j'th "left singular vector" of A
v_j is the j'th "right singular vector" of A.

Then:

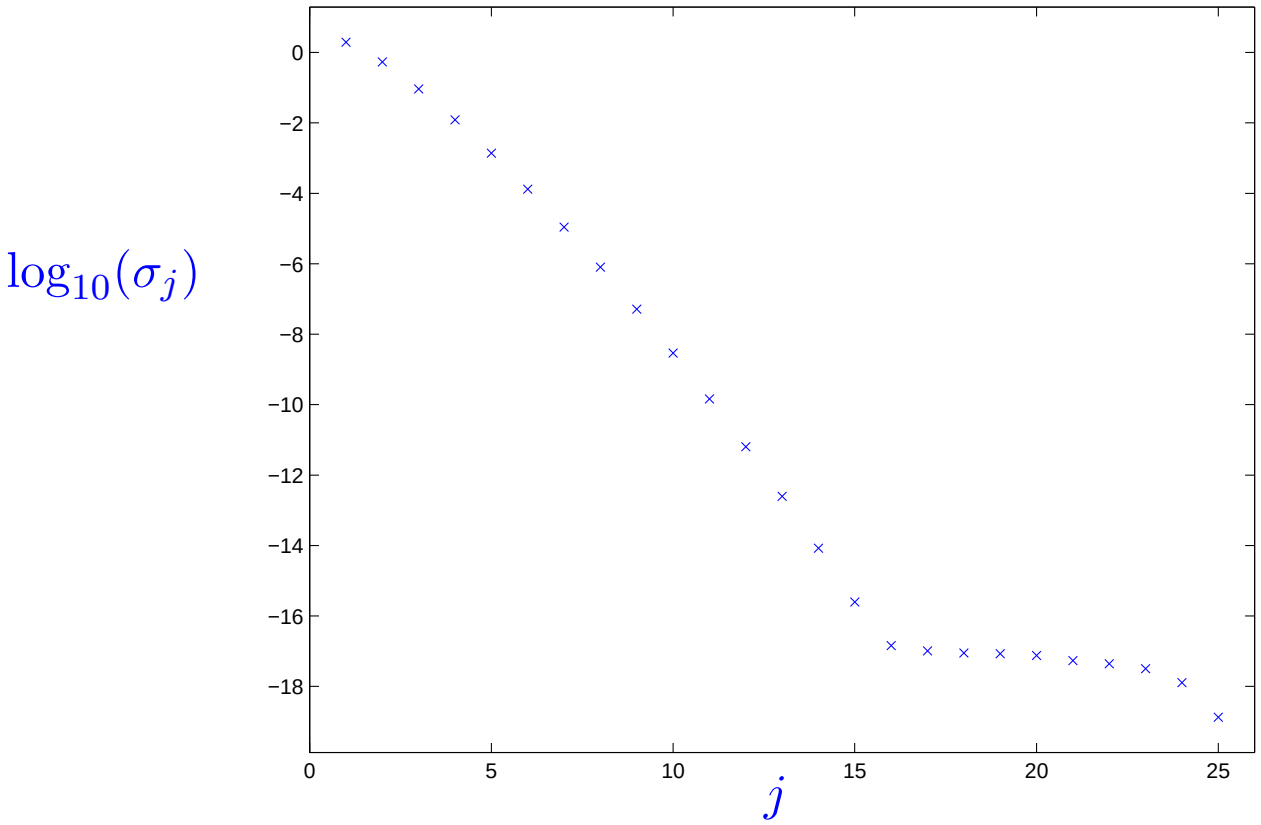
$$\sigma_j = \inf_{\mathrm{rank}(\tilde{A})=j-1} ||A - \tilde{A}||.$$

and

$$\mathrm{argmin}\{||A - \tilde{A}|| :\ \mathrm{rank}(\tilde{A}) = k\} = \sum_{j=1}^{k} \sigma_j\, u_j\, v_j^{\mathrm{t}}.$$

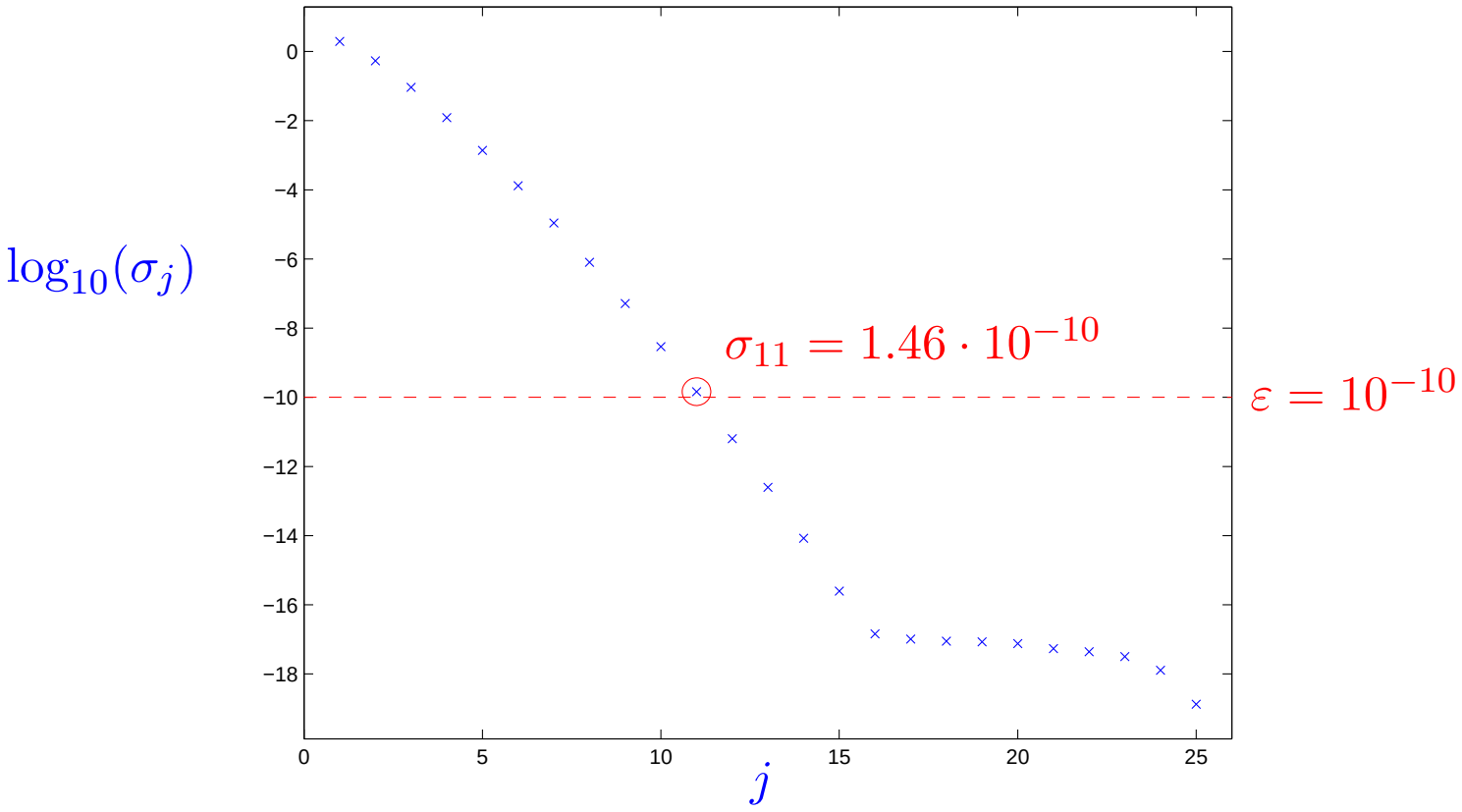
The decay of the singular values determines how well a matrix can be approximated by low-rank factorizations.

Example: Let A be the 25×25 Hilbert matrix, *i.e.* $A_{ij} = 1/(i+j-1)$. Let σ_j denote the j'th singular value of A.



The decay of the singular values determines how well a matrix can be approximated by low-rank factorizations.

Example: Let A be the 25×25 Hilbert matrix, *i.e.* $A_{ij} = 1/(i+j-1)$. Let σ_j denote the j'th singular value of A.



For instance, to precision $\varepsilon = 10^{-10}$, the matrix A has rank 11.

Model problem: Find an approximate basis for the column space of a given matrix.	• Let ε denote the computational accuracy desired. • Let A be an $N \times N$ matrix. • Determine an integer k and an $N \times k$ ON-matrix Q such that $\|\|A - Q\,Q^{\mathrm{t}}\,A\|\| \leq \varepsilon$.

Model problem: Find an approximate basis for the column space of a given matrix.	• Let ε denote the computational accuracy desired. • Let A be an $N \times N$ matrix. • Determine an integer k and an $N \times k$ ON-matrix Q such that $\|\|A - Q\,Q^{\mathrm{t}}\,A\|\| \leq \varepsilon$.

Notes:

- Once Q has been constructed, it is in many environments possible to construct standard factorization (such as the SVD/PCA) using $O(N\,k^2)$ operations. Specifically, this is true if either
  - matrix vector products $x \mapsto x'\,A$ can be evaluated rapidly, or,
  - individual entries of A can be computed in $O(1)$ operations.
- We seek a k that is as small as possible, but it is not a priority to make it absolutely optimal.
- If the Q initially constructed has too many columns, but is accurate, then the true optimal rank is revealed by postprocessing.

Model problem: Find an approximate basis for the column space of a given matrix.	• Let ε denote the computational accuracy desired. • Let A be an $N \times N$ matrix. • Determine an integer k and an $N \times k$ ON-matrix Q such that $\|\|A - Q\,Q^{\mathrm{t}}\,A\|\| \leq \varepsilon$.

We will discuss two environments:

Case 1:

We have a fast technique for evaluating matrix-vector products. Let T_{mult} denote the cost. We assume $T_{\text{mult}} \ll N^2$.

Standard methods (*e.g.* Lanczos) require $O(T_{\text{mult}}\,k)$ operations.

The new methods are also $O(T_{\text{mult}}\,k)$ but are more robust, (more accurate,) and better suited for parallelization.

Case 2:

A is a general $N \times N$ matrix.

Standard methods (*e.g.* Gram-Schmidt) require $O(N^2\,k)$ operations.

The new method requires $O(N^2\,\log(k))$ operations.

The methods that we propose are based on **randomized sampling.**

This means that they have a non-zero probability of giving an answer that is not accurate to within the requested accuracy.

The probability of failure can be balanced against computational cost by the user.

It can very cheaply be rendered entirely negligible; failure probabilities less than 10^{-15} are standard. (In other words, if you computed 1 000 matrix factorizations a second, you would expect to see one "failure" every 30 000 years.)

Definition: We say that an $m \times n$ matrix Ω is a Gaussian random matrix if

$$\Omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & \vdots & & \vdots \\ \omega_{m1} & \omega_{m2} & \cdots & \omega_{mn} \end{bmatrix},$$

where the numbers ω_{ij} are random variables drawn independently from a normalized Gaussian distribution.

Note: The probability distribution of Ω is isotropic in the sense that if $U \in \mathrm{O}(m)$ and $V \in \mathrm{O}(n)$, then $U\,\Omega\,V$ has the same distribution as Ω.

Note: In practise, the random numbers used will be constructed using "random number generators." The quality of the generators will not matter much. (Shockingly little, in fact.)

We start with Case 1: We know how to compute the product $x \mapsto A\,x$ rapidly.

Algorithm 1: *Rapid computation of a low-rank approximation.*	• Let ε denote the computational accuracy desired. • Let A be an $N \times N$ matrix of ε-rank k. • We seek a basis for $\mathrm{Col}(A)$. • We can perform matrix-vector multiplies fast.

1. Fix a small positive integer p (representing how much "oversampling" we do). Construct a Gaussian random matrix Ω of size $n \times (k+p)$.

2. Form the $m \times (k+p)$ matrix $Y = A\,\Omega$.

3. Construct an $m \times (k+p)$ orthogonal matrix Q such that $Y = Q\,Q^{\mathrm{t}}\,Y$.

Each column of Y is a sample from the column space of A.
The more samples we have, the more likely it is that

$$||A - Q\,Q^{\mathrm{t}}\,A|| \leq \varepsilon. \tag{1}$$

If we were very lucky, then (1) would hold with $p = 0$.

Question: How large does p need to be in practice?

How to measure "how well we are doing":

Let Ω_ℓ be a Gaussian random matrix of size $n \times \ell$.

Set $Y_\ell = [y_1, y_2, \ldots, y_\ell] = A\,\Omega_\ell$.

Let Q_ℓ be an $m \times \ell$ matrix such that $Y_\ell = Q_\ell\, Q_\ell^{\mathrm{t}}\, Y_\ell$.

The "error" after ℓ steps is then

$$e_\ell = ||A - Q_\ell\, Q_\ell^{\mathrm{t}}\, A||$$

The quantity e_ℓ should be compared to the minimal error

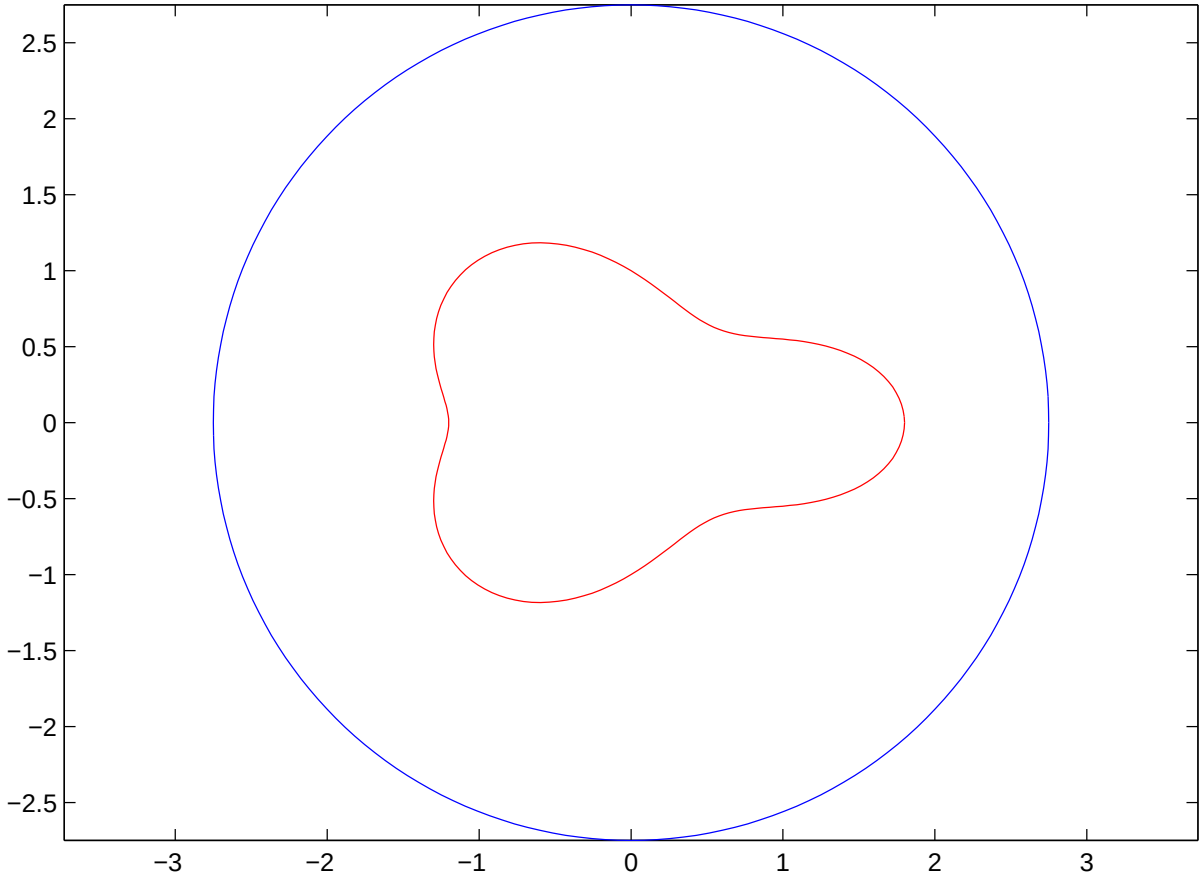
$$\sigma_{\ell+1} = \min_{\mathrm{rank}(B)=\ell} ||A - B||.$$

Specific example to illustrate the performance:

Let A be a 200×200 matrix arising from discretization of

$$[\mathcal{S}_{\Gamma_1 \leftarrow \Gamma_2}\, u](x) = \alpha \int_{\Gamma_2} \log|x-y|\, u(y)\, ds(y), \qquad x \in \Gamma_1,$$

where Γ_1 is shown in red and Γ_2 is shown in blue:

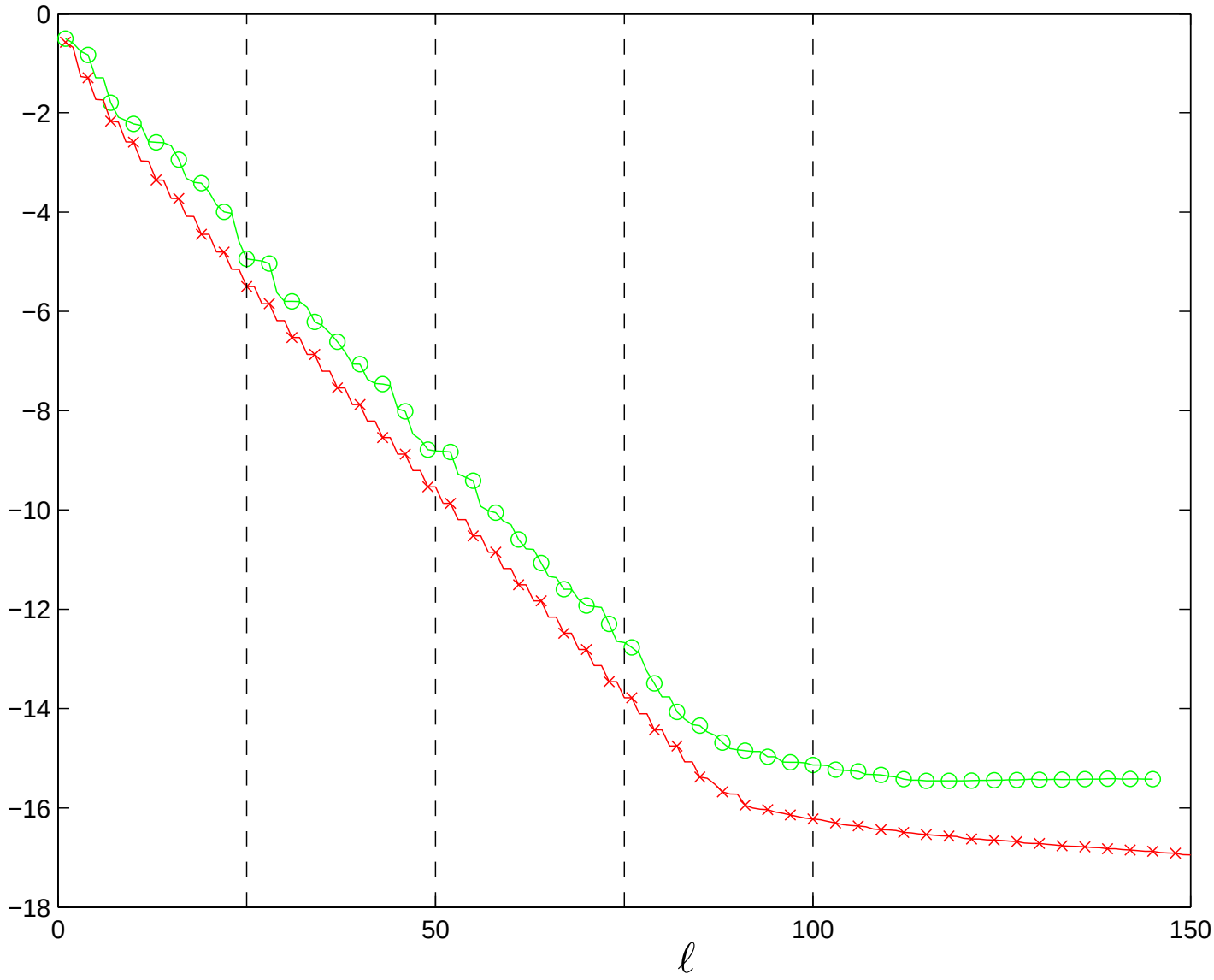


The number α is chosen so that $||A|| = \sigma_1 = 1$.

Results from one realization of the randomized algorithm

$\log_{10}(e_\ell)$
(actual error)

$\log_{10}(\sigma_{\ell+1})$
(theoretically
minimal error)



How to measure "how well we are doing" — revisited:

Let Ω_ℓ be a Gaussian random matrix of size $n \times \ell$.

Set $Y_\ell = [y_1, y_2, \ldots, y_\ell] = A\,\Omega_\ell$.

Let Q_ℓ be an $m \times \ell$ matrix such that $Y_\ell = Q_\ell\, Q_\ell^{\mathrm{t}}\, Y_\ell$.

The "error" after ℓ steps is then

$$e_\ell = ||A - Q_\ell\, Q_\ell^{\mathrm{t}}\, A||$$

The quantity e_ℓ should be compared to the minimal error

$$\sigma_{\ell+1} = \min_{\mathrm{rank}(B)=\ell} ||A - B||.$$

How to measure "how well we are doing" — revisited:

Let Ω_ℓ be a Gaussian random matrix of size $n \times \ell$.

Set $Y_\ell = [y_1, y_2, \ldots, y_\ell] = A\,\Omega_\ell$.

Let Q_ℓ be an $m \times \ell$ matrix such that $Y_\ell = Q_\ell\, Q_\ell^{\mathrm{t}}\, Y_\ell$.

The "error" after ℓ steps is then

$$e_\ell = ||A - Q_\ell\, Q_\ell^{\mathrm{t}}\, A||$$

The quantity e_ℓ should be compared to the minimal error

$$\sigma_{\ell+1} = \min_{\mathrm{rank}(B)=\ell} ||A - B||.$$

In reality, computing e_ℓ is not affordable. Instead, we compute something like

$$f_\ell = \max_{1 \le j \le 10} \left|\left| \left(I - Q_\ell\, Q_\ell^{\mathrm{t}}\right) y_{l+j} \right|\right|.$$

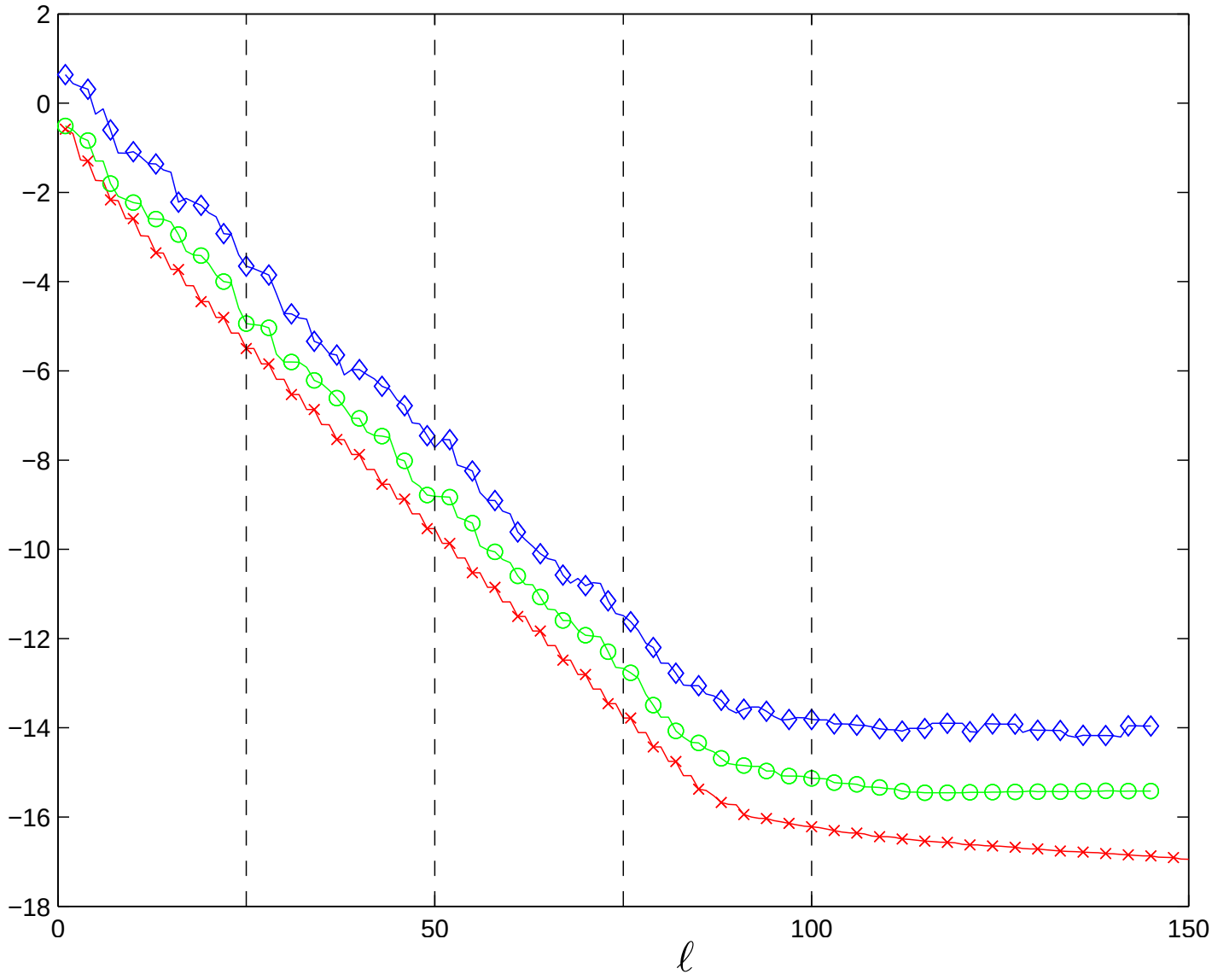
The computation stops when we come to an ℓ such that $f_\ell < \varepsilon \times$ [constant].

Results from one realization of the randomized algorithm

$\log_{10}(10\, f_\ell)$
(error bound)

$\log_{10}(e_\ell)$
(actual error)

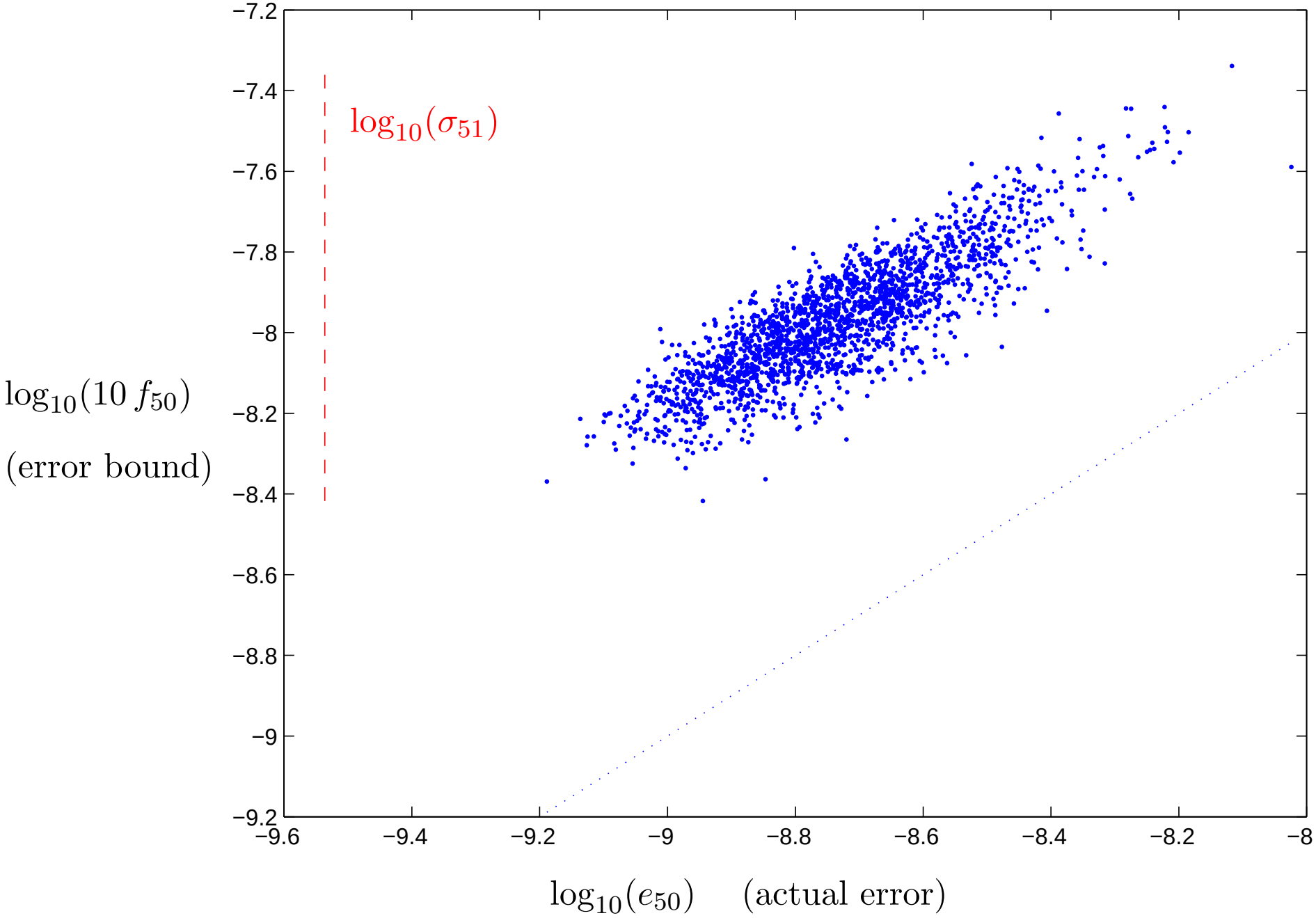
$\log_{10}(\sigma_{\ell+1})$
(theoretically
minimal error)



Note: The development of an error estimator resolves the issue of not knowing the numerical rank in advance!

Was this just a lucky realization?

Each dots represents one realization of the experiment with $k = 50$ samples:



Important:

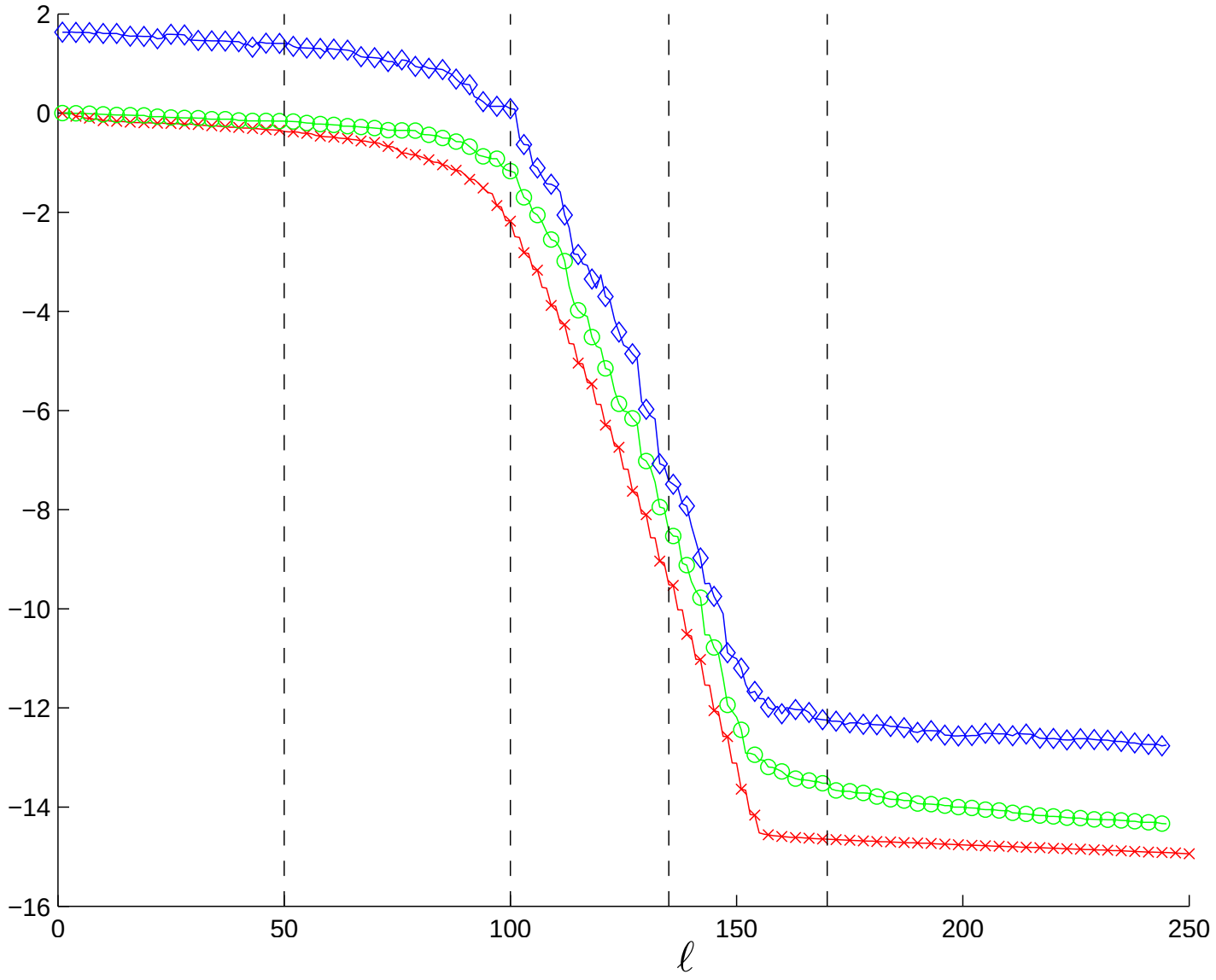
- What is stochastic is the *run time*, not the accuracy.
- The error in the factorization is (practically speaking) always within the prescribed tolerance.
- Post-processing (practically speaking) always determines the rank correctly.

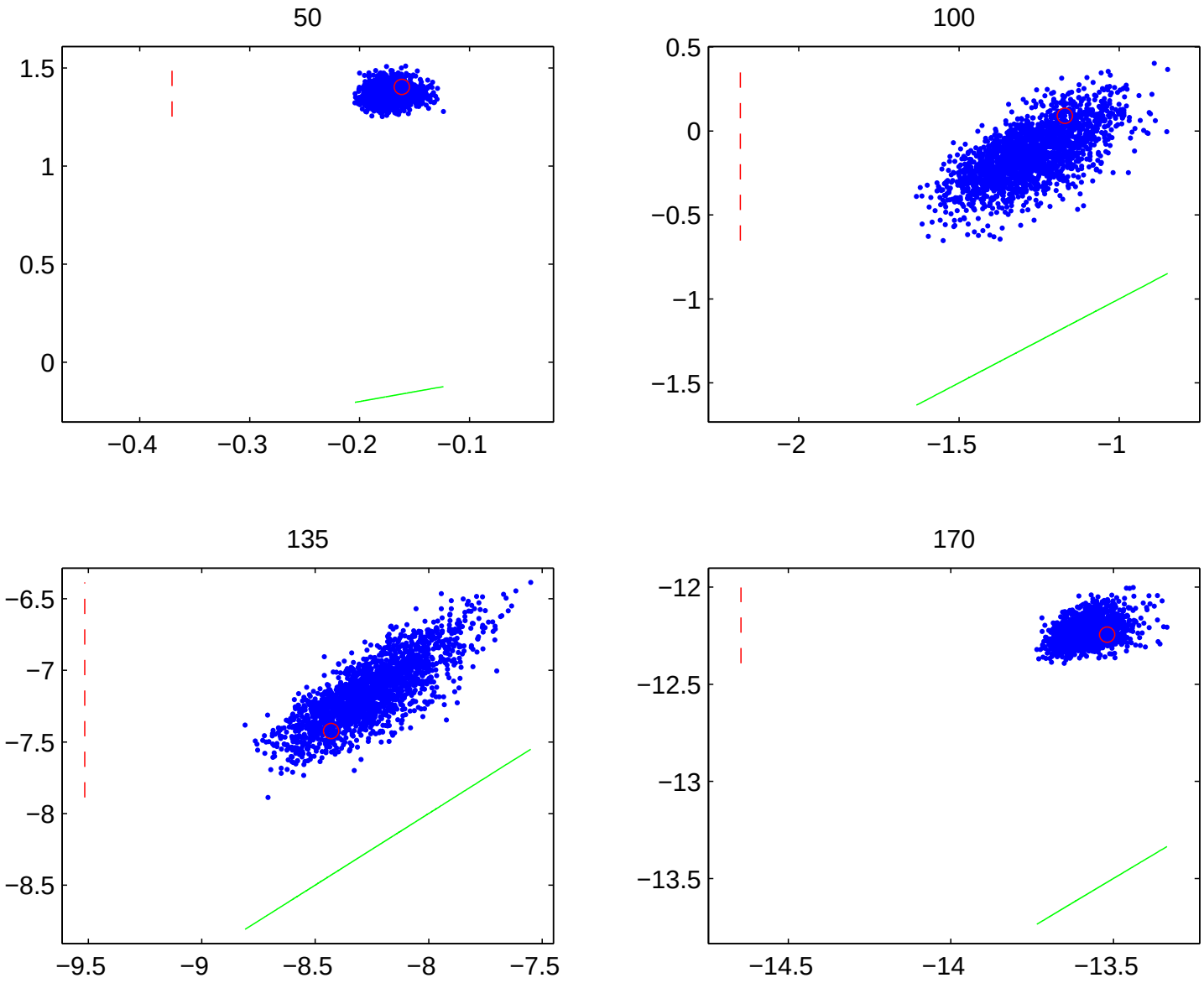
Results from a high-frequency Helmholtz problem (complex arithmetic)

$\log_{10}(10\, f_\ell)$
(error bound)

$\log_{10}(e_\ell)$
(actual error)

$\log_{10}(\sigma_{\ell+1})$
(theoretically
minimal error)




50
1.5
1
0.5
0
−0.4
−0.3
−0.2
−0.1
100
0.5
0
−0.5
−1
−1.5
−2
−1.5
−1
135
−6.5
−7
−7.5
−8
−8.5
−9.5
−9
−8.5
−8
−7.5
170
−12
−12.5
−13
−13.5
−14.5
−14
−13.5

So far, we have assumed that we have a fast matrix-vector multiplier at our disposal.

What happens if we do not?

So far, we have assumed that we have a fast matrix-vector multiplier at our disposal.

What happens if we do not?

In this case, $T_{\text{mult}} = N^2$ so the computational cost of Algorithm I is

$$O(T_{\text{mult}}\, k + N\, k^2) = O(N^2\, k + N\, k^2).$$

When $k \ll N$, Algorithm 1 might be slightly faster than Gram-Schmidt:

Multiplications required for Algorithm 1:	$N^2\,(k+10)$	$+\,O(k^2\,N)$
Multiplications required for Gram-Schmidt:	$N^2\,2\,k$	

Other benefits (sometimes more important ones than CPU count):

- Data-movement.
- Parallelization.
- More accurate. (This requires some additional twists not yet described.)

However, many environments remain in which there is little or no gain.

Algorithm 2: An $O(N^2 \log(k))$ algorithm for *general* matrices:

Proposed by Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert.

Recall that Algorithm 1 determines a basis for the column space from the matrix

$$\underset{N \times \ell}{Y} = \underset{N \times N}{A} \; \underset{N \times \ell}{\Omega}.$$

Key points:

- The entries of Ω are i.i.d. random numbers.
- The product $x \mapsto A\,x$ can be evaluated rapidly.

What if we do *not* have a fast algorithm for computing $x \mapsto A\,x$?

New idea: Construct Ω with "some randomness" and "some structure".
Then for each $1 \times N$ row a of A, the matrix-vector product

$$a \mapsto a\,\Omega$$

can be evaluated using $N \log(\ell)$ operations.

What is this "random but structured" matrix Ω?

$$\begin{array}{ccccc} \Omega & = & D & F & S \\ N \times \ell & & N \times N & N \times N & N \times \ell \end{array}$$

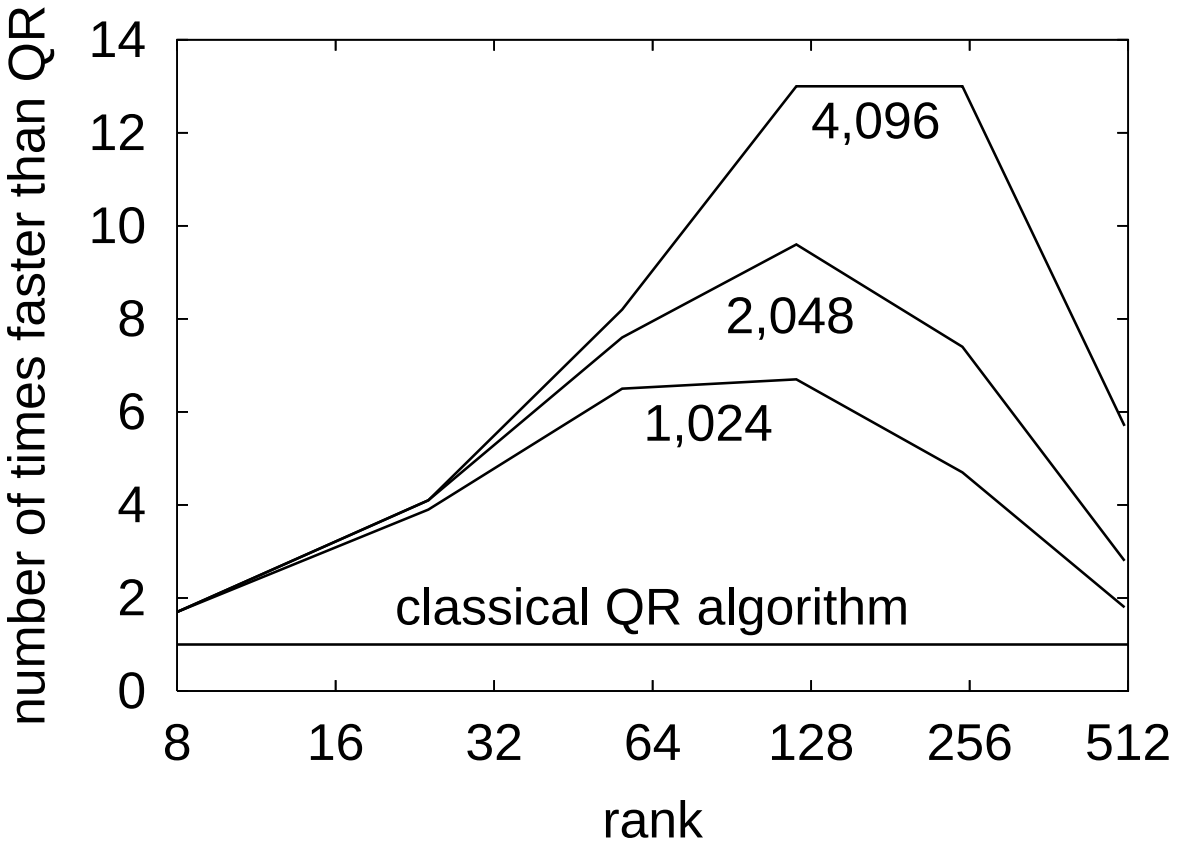
where,

- D is a diagonal matrix whose entries are i.i.d. random variables drawn from a uniform distribution on the unit circle in $\mathbb{C}$.
- F is the discrete Fourier transform, $F_{jk} = \dfrac{1}{N^{1/2}}\, e^{-2\pi i (j-1)(k-1)/N}$.
- S is a matrix whose entries are all zeros except for a single, randomly placed 1 in each column. (In other words, the action of S is to draw ℓ columns at random from $D\,F$.)

Note: Other successful choices of the matrix Ω have been tested, for instance, the Fourier transform may be replaced by the Walsh-Hadamard transform.

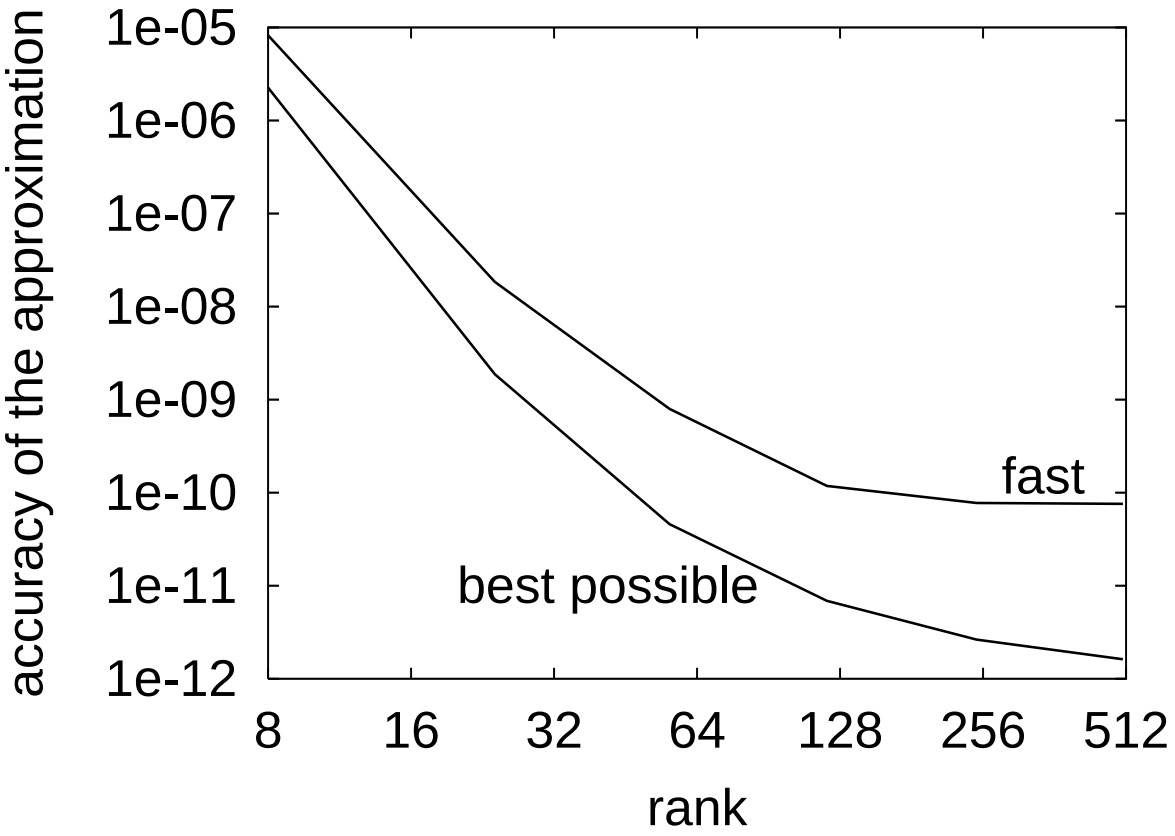
This idea was described by Nir Ailon and Bernard Chazelle (2006).
There is also related recent work by Sarlós (on randomized regression).

Speed gain on square matrices of various sizes



The time required to verify the approximation is included in the fast, but not in the classical timings.

Empirical accuracy on 2,048-long convolution



The estimates of the accuracy of the approximation are accurate to at least two digits of relative precision.

The accuracy of the randomized method has recently been improved.

Theory / context

In the remainder of the talk we will focus on Algorithm 1 (for the case when fast matrix-vector multiplies are available). For this case, we have fairly sharp estimates of the "failure probabilities."

The theoretical results to be presented are related to (and in some cases inspired by) earlier work on randomized methods in linear algebra. This work includes:

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 (2006a, 2006b, 2006c, 2006d)

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)

Theorem (Martinsson, Rokhlin, Tygert 2006):

Let A be an $M \times N$ matrix.

Let k and p be positive integers. (k is "rank" and p is the degree of "oversampling")

Let Ω be an $N \times (k+p)$ Gaussian random matrix.

Let Q be an $M \times (k+p)$ matrix whose columns form an ON-basis for the columns of $A\,\Omega$.

Set $\sigma_{k+1} = \min_{\mathrm{rank}(B)=k} ||A - B||.$

Then

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\;\sigma_{k+1},$$

with probability at least

$$1 - \varphi(p),$$

where φ is a decreasing function satisfying, e.g.,

$\varphi(5) < 3 \cdot 10^{-6}, \qquad \varphi(10) < 3 \cdot 10^{-13}, \qquad \varphi(15) < 8 \cdot 10^{-21}, \qquad \varphi(20) < 6 \cdot 10^{-27}.$

Theorem (Martinsson, Rokhlin, Tygert 2006):

Let A be an $M \times N$ matrix.

Let k and p be positive integers. (k is "rank" and p is the degree of "oversampling")

Let Ω be an $N \times (k+p)$ Gaussian random matrix.

Let Q be an $M \times (k+p)$ matrix whose columns form an ON-basis for the columns of $A\,\Omega$.

Set $\sigma_{k+1} = \min_{\text{rank}(B)=k} ||A - B||.$

Then

$$||A - Q\,Q^{\text{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\,\sigma_{k+1}, \qquad \leftarrow \textit{Not good!}$$

with probability at least

$$1 - \varphi(p),$$

where φ is a decreasing function satisfying, e.g.,

$$\varphi(5) < 3 \cdot 10^{-6}, \qquad \varphi(10) < 3 \cdot 10^{-13}, \qquad \varphi(15) < 8 \cdot 10^{-21}, \qquad \varphi(20) < 6 \cdot 10^{-27}.$$

The key bound in the proof is the line:

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\,\sigma_{k+1}.$$

The factor in blue represents the degree of suboptimality.

In applications where the singular values decay rapidly, this factor does not represent a problem. (A slight increase in k kills off the factor.)

The key bound in the proof is the line:

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\,\sigma_{k+1}.$$

The factor in blue represents the degree of suboptimality.

In applications where the singular values decay rapidly, this factor does not represent a problem. (A slight increase in k kills off the factor.)

However, in many applications of interest, the entries of A may be very noisy, and it may be that

$$\sigma_{k+1} \approx \sigma_{k+2} \approx \cdots \approx \sigma_N \approx 10^{-1} \times \sigma_1.$$

The key bound in the proof is the line:

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\,\sigma_{k+1}.$$

The factor in blue represents the degree of suboptimality.

In applications where the singular values decay rapidly, this factor does not represent a problem. (A slight increase in k kills off the factor.)

However, in many applications of interest, the entries of A may be very noisy, and it may be that

$$\sigma_{k+1} \approx \sigma_{k+2} \approx \cdots \approx \sigma_N \approx 10^{-2} \times \sigma_1.$$

Moreover, N may be very large — $N \sim 10^5$ — $N \sim 10^8$ $\cdots$

The key bound in the proof is the line:

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{(k+p)\,(N-k)}\,\sigma_{k+1}.$$

The factor in blue represents the degree of suboptimality.

In applications where the singular values decay rapidly, this factor does not represent a problem. (A slight increase in k kills off the factor.)

However, in many applications of interest, the entries of A may be very noisy, and it may be that

$$\sigma_{k+1} \approx \sigma_{k+2} \approx \cdots \approx \sigma_N \approx 10^{-2} \times \sigma_1.$$

Moreover, N may be very large — $N \sim 10^5$ — $N \sim 10^8$ $\cdots$

The suboptimality is damning in data mining and signal processing applications. Here we have HUGE matrices, and lots of noise.

Theorem: *[Halko, Martinsson, Tropp 2009] Fix a* real $m \times n$ *matrix* A *with singular values* $\sigma_1, \sigma_2, \sigma_3, \ldots$. *Choose integers* $k \geq 1$ *and* $p \geq 2$, *and draw an* $n \times (k+p)$ *standard Gaussian test matrix* Ω. *Construct the data matrix* $Y = A\Omega$, *and let* P_Y *denote the orthogonal projection onto the range of* Y. *Then*

$$\mathbb{E}\|(I - P_Y)A\|_{\mathrm{F}} \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\infty} \sigma_j^2\right)^{1/2}.$$

Moreover,

$$\mathbb{E}\|(I - P_Y)A\| \leq \left(1 + \sqrt{\frac{k}{p-1}}\right) \sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \left(\sum_{j=k+1}^{\infty} \sigma_j^2\right)^{1/2}.$$

- Numerical experiments indicate that these estimates are close to sharp.
- When $\sigma_j \sim \beta^j$, we have $\left(\sum_{j=k+1}^{\infty} \sigma_j^2\right)^{1/2} \sim \sigma_{k+1} \frac{1}{1-\beta}$.
- Tail probabilities are often in a practical sense irrelevant.

Power method for improving accuracy:

Note that the error depends on how quickly the singular values decay.

The faster the singular values decay — the higher the relative weight of the dominant modes are weighted in the samples.

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

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

Much faster decay — so use the sample matrix

$$Z = B\,\Omega = (A\,A^*)^q\,A\,\Omega$$

instead of

$$Y = A\,\Omega.$$

Power method for improving accuracy:

The following theorem is inspired by results by Rokhlin, Szlam, and Tygert (2008):

Theorem: *[Halko, Martinsson, Tropp 2009] Let m, n, and ℓ be positive integers such that $\ell < n \leq m$. Let A be an $m \times n$ matrix and let Ω be an $n \times \ell$ matrix. Let q be a non-negative integer, set $B = (A^* A)^q A$, and construct the sample matrix $Z = B\,\Omega$. Then*

$$|||(I - P_Z)\,A||| \leq |||(I - P_Z)\,B|||^{1/(2q+1)}$$

where $|||\cdot|||$ denotes either the spectral norm or the Frobenius norm.

Since the ℓ'th singular value of $B = (A^* A)^q A$ is $\sigma_\ell^{2\,q+1}$, any result of the type

$$||(I - P_Y)\,A|| \leq C\,\sigma_{k+1},$$

where $Y = A\,\Omega$ and $C = C(m, n, k)$, gets improved to a result

$$||(I - P_Z)\,A|| \leq C^{1/(2\,q+1)}\,\sigma_{k+1}$$

when $Z = (A^* A)^q A\,\Omega$.

10-log of errors

incurred when using

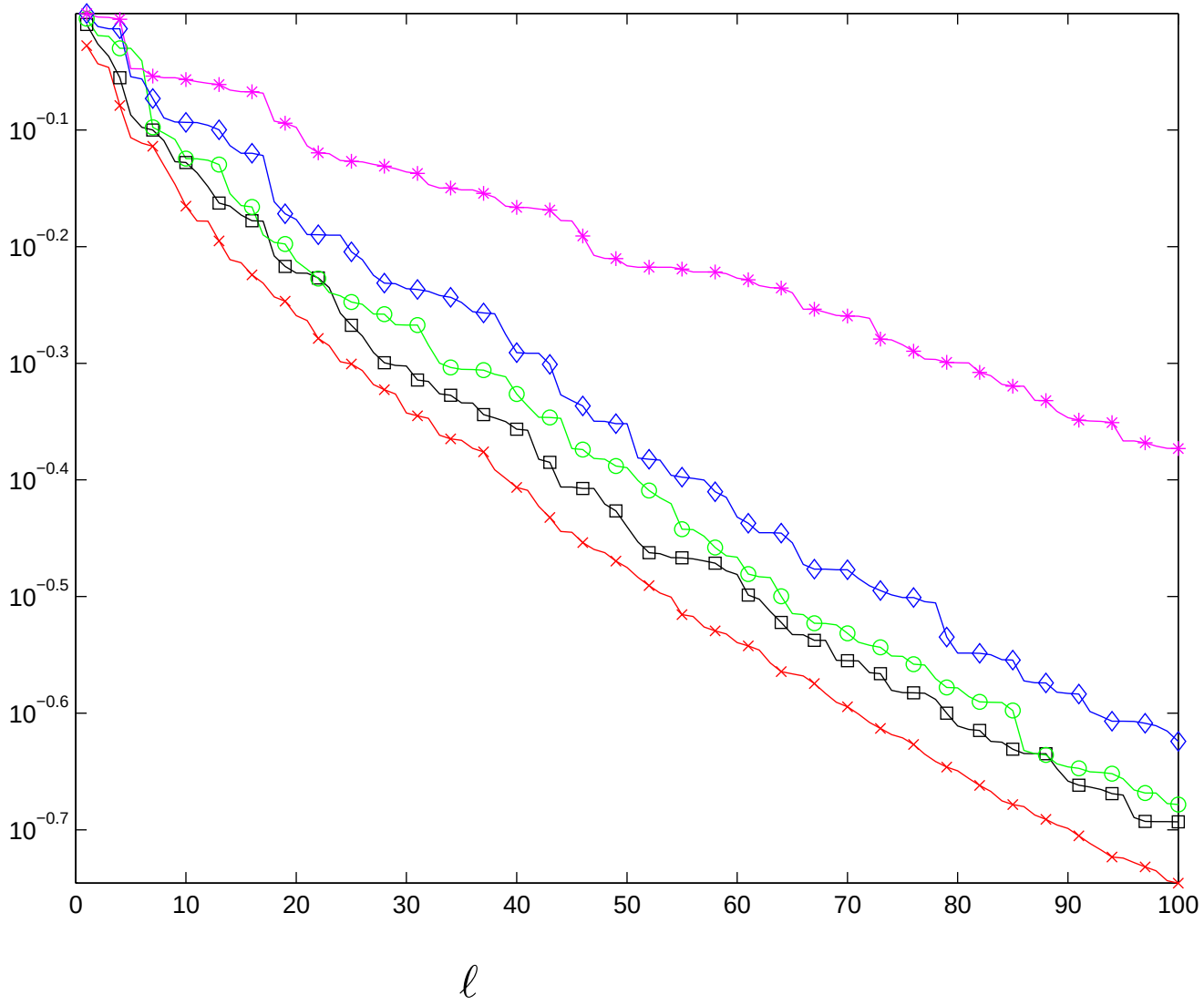
the power method with:

$q = 0$ in pink

$q = 1$ in blue

$q = 2$ in green

$q = 3$ in black



The matrix A being analyzed is a 9025×9025 matrix arising in image processing.
(It is a graph Laplacian on the manifold of 9×9 patches.)
The red crosses mark the singular values of A.

Some observations ...

The observation that a "thin" Gaussian random matrix to high probability is well-conditioned is at the heart of the celebrated Johnson-Lindenstrauss lemma:

Lemma: *Let ε be a real number such that $\varepsilon \in (0, 1)$, let n be a positive integer, and let k be an integer such that*

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

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

$$(1 - \varepsilon) \, ||u - v||^2 \leq ||f(u) - f(v)|| \leq (1 + \varepsilon) \, ||u - v||^2, \qquad \forall \; u, v \in V. \tag{3}$$

Further, such a map can be found in randomized polynomial time.

It has been shown that an excellent choice of the map f is the linear map whose coefficient matrix is a $k \times d$ matrix whose entries are i.i.d. Gaussian random variables (see, *e.g.* Dasgupta & Gupta (1999)).
When k satisfies, (2), this map satisfies (3) with probability close to one.

The related Bourgain embedding theorem shows that such statements are not restricted to Euclidean space:

Theorem:. *Every finite metric space (X, d) can be embedded into ℓ^2 with distortion $O(\log n)$ where n is the number of points in the space.*

Again, random projections can be used as the maps.

The Johnson-Lindenstrauss lemma (and to some extent the Bourgain embedding theorem) expresses a theme that is recurring across a number of research areas that have received much attention recently. These include:

- Compressed sensing (Candès, Tao, Romberg, Donoho).
- Approximate nearest neighbor search (Jones, Rokhlin).
- Geometry of point clouds in high dimensions (Coifman, Jones, Lafon, Lee, Maggioni, Nadler, Singer, Warner, Zucker, *etc*).
- Construction of multi-resolution SVDs.
- Clustering algorithms.
- Search algorithms / knowledge extraction.

Note: Omissions! No ordering. Missing references. Etc etc.

Many of these algorithms work "unreasonably well."

The randomized algorithm presented here is close in spirit to randomized algorithms such as:

- Randomized quick-sort.
  (With variations: computing the median / order statistics / *etc.*)
- Routing of data in distributed computing with unknown network topology.
- Rabin-Karp string matching / verifying equality of strings.
- Verifying polynomial identities.

Many of these algorithms are of the type that it is the *running time* that is stochastic. The quality of the final output is excellent.

The randomized algorithm that is perhaps the best known within numerical analysis is Monte Carlo. This is somewhat lamentable given that MC is often a "last resort" type algorithm used when the curse of dimensionality hits — inaccurate results are tolerated simply because there are no alternatives. (These comments apply to the traditional "unreformed" version of MC — for many applications, more accurate versions have been developed.)

Observation: Mathematicians working on these problems often focus on minimizing the distortion factor

$$\frac{1+\varepsilon}{1-\varepsilon}$$

arising in the Johnson-Lindenstrauss bound:

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

In our environments, we do not need this constant to be particularly close to 1. It should just not be "large" — say less that 10 or some such.

This greatly reduces the number of random projections needed! Recall that in the Johnson-Lindenstrauss theorem:

$$\text{number of samples required } \sim \frac{1}{\varepsilon^2}\,\log(N).$$

Observation: Multiplication by a random unitary matrix reduces any matrix to its "general" form. All information about the singular vectors vanish. (The singular *values* remain the same.)

This opens up the possibility for general pre-conditioners —
counterexamples to various algorithms can be disregarded.

The feasibility has been demonstrated for the case of least squares solvers for very large, very over determined systems. (Work by Rokhlin & Tygert, Sarlós, ...)

Work on $O(N^2 \, (\log N)^2)$ solvers of general linear systems is under way.
(Random pre-conditioning + iterative solver.)

May stable fast matrix inversion schemes for general matrices be possible?

Observation: Robustness with respect to the quality of the random numbers.

The assumption that the entries of the random matrix are i.i.d. normalized Gaussians simplifies the analysis since this distribution is invariant under unitary maps.

In practice, however, one can use a low quality random number generator. The entries can be uniformly distributed on $[-1, 1]$, they be drawn from certain Bernouilli-type distributions, *etc.*

Remarkably, they can even have enough internal structure to allow fast methods for matrix-vector multiplications. For instance:

- Subsampled discrete Fourier transform.
- Subsampled Walsh-Hadamard transform.
- Givens rotations by random angles acting on random indices.

This was exploited in "Algorithm 2" (and related work by Ailon and Chazelle).
Our theoretical understanding of such problems is unsatisfactory.
Numerical experiments perform *far* better than existing theory indicates.

Even though it is thorny to *prove* some of these results (they draw on techniques from numerical analysis, probability theory, functional analysis, theory of randomized algorithms, *etc*), work on randomized methods in linear algebra is progressing fast.

Important: Computational prototyping of these methods is extremely simple.

- Simple to code an algorithm.
- They work so well that you immediately know when you get it right.

Current research directions:

- Acceleration of BLAS / LINPACK functions.
  - May actually soon be integrated in Matlab and Mathematica.

- Construction of reduced models for physical phenomena ("model reduction").
  - Wave propagation through media with periodic micro-structures.
  - Scattering problems involving multiple scatterers.

- New estimates on spectral properties of random matrices.

- Acceleration of fast matrix algorithms such as the "Fast Multipole Method".

- Approximation of very large very noisy data sets stored on disk or streamed.

  The randomized algorithms solve two fundamental limitations of existing methods:
  - Propagation of rounding errors.
  - Very few passes over data. (Sometimes only one!)

  Important applications that cannot be solved with existing technology:
  Image and video processing / network analysis / statistical data processing / ...