Fast matrix computations via randomized sampling

Per-Gunnar Martinsson
The University of Colorado at Boulder

Students:

Tracy Babb

JaeAnn Dwulet

Adrianna Gillman

Nathan Halko

Patrick Young

Collaborators:

Edo Liberty (Yale)

Vladimir Rokhlin (Yale)

Yoel Shkolnisky (Yale)

Joel Tropp (Caltech)

Mark Tygert (Courant)

Franco Woolfe (Goldman Sachs)

Reference: *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions* N. Halko, P.G. Martinsson, J. Tropp. Sep. 2009.

Low-rank approximation

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

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

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

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

Real life is messy — Part I:

In applications, the form of the factors typically matters.

The *Singular Value Decomposition* (SVD)

$$\begin{array}{ccccc} A & = & U & \Sigma & V^* \\ N \times N & & N \times k & k \times k & k \times N \\ & & \text{orthonormal} & \text{diagonal} & (\text{orthonormal})^* \end{array}$$

is important in many applications. For instance:

- Principal Component Analysis (PCA).
- Extracting information from large graphs.
- “Eigenfaces”.
- *etc*

Real life is messy — Part I:

In applications, the form of the factors typically matters.

The *Interpolatory Decomposition* (ID) determines a spanning set of columns:

$$\begin{array}{cccc} A & = & A_{\text{col}} & X \\ N \times N & & N \times k & k \times N \\ & & \text{subset} & \\ & & \text{of columns} & \end{array}$$

Applications in:

- Sparse representation of data.
- Analysis of genetic and financial data sets.
- Fast numerical algorithms (such as the Fast Multipole Method).
- *etc*

Real life is messy — Part I:

In applications, the form of the factors typically matters.

The *QR decomposition*

$$\begin{array}{ccccc} A & = & Q & R & P \\ N \times N & & N \times k & k \times N & N \times N \\ & & \text{orthonormal} & \text{upper triangular} & \text{permutation} \end{array}$$

Applications in:

- Numerical analysis.
- Solving least squares problems.
- *etc*

Real life is messy — Part II:

In practice, we very rarely are lucky enough to work with matrices of exact rank k. Instead, we have to consider 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

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

Add conditions on the form of $\tilde{A}$ and things start to get complicated ...

On a well-known list[a] of the “Top 10 Algorithms” that have influenced the practice of science and engineering during the 20th century, we find an entry that is not really an algorithm: the idea of using matrix factorizations to accomplish basic tasks in numerical linear algebra:

> *The underlying principle of the decompositional approach to matrix computation is that it is not the business of the matrix algorithmicists to solve particular problems but to construct computational platforms from which a variety of problems can be solved.*

The lesson for us is: We need to formulate a primitive problem that forms a reusable building block in all these applications.

[a] J. Dongarra and F.Sullivan, *The top 10 algorithms*, Comput. Sci. Eng., (2000), pp. 22–23.

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

In other words, the columns of Q form an ON-basis for the range of A.

We want ℓ to be reasonably close to the theoretically minimal number, but it is more important for the approximation to be accurate.

We for now assume that we know the approximate rank in advance; and will deal with the case where we do not later.

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Claim: If a Q such that $A \approx Q\, Q^*\, A$ is provided, then any standard factorization of A can be computed inexpensively.

How to construct the SVD of A:

1. Form the matrix $B = Q^*\, A$.

2. Compute the SVD of B so that $B = \hat{U}\, \Sigma\, V^*$.

3. Compute the product $U = Q\, \hat{U}$.

The output U, Σ, V of this process satisfies $||A - U\, \Sigma\, V^*|| = ||A - Q\, Q^* A||$.

The given process requires $O(mn\ell)$ flops. This can be reduced to $O((m+n)\ell^2)$.

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Claim: If a Q such that $A \approx Q\, Q^*\, A$ is provided, then any standard factorization of A can be computed inexpensively.

How to construct an Interpolatory Decomposition of A:

1. Find ℓ spanning rows of Q so that $Q = X\, Q(J, :\ )$.

2. Now $A = X\, A(J, :\ )$ automatically!

This process costs only $O(m\,\ell^2)$ flops, but leads to some loss of accuracy:

$$||A - A(J, :\ )\, X|| \leq \big(1 + \sqrt{1 + 4\,\ell\,(n - \ell)}\big)\, ||A - Q\,Q^*A||.$$

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Claim: If a Q such that $A \approx Q\, Q^*\, A$ is provided, then any standard factorization of A can be computed inexpensively.

How to construct the SVD *via* the Interpolatory Decomposition:

1. Find ℓ spanning rows of Q so that $Q = X\, Q(J,:)$.
2. Compute the QR factorization of X so that $X = \hat{U}\, R$ where $\hat{U}$ is orthonormal.
3. Form the product $R\, A(J,:)$ and comute its SVD: $R\, A(J:) = \tilde{U}\, \Sigma\, V^*$.
4. Set $U = \hat{U}\, \tilde{U}$.

This process costs only $O((m+n)\,\ell^2)$ flops, but leads to some loss of accuracy:

$$||A - U\,\Sigma\,V^*|| \leq \left(1 + \sqrt{1 + 4\,\ell\,(n-\ell)}\right)\, ||A - Q\,Q^*A||.$$

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^{*}\, A$.

What is the theoretically optimal solution?

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Recall that any matrix A has a singular value decomposition (SVD):

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

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

$$\sigma_{\ell+1} = \inf\{||A - \tilde{A}|| :\ \text{rank}(\tilde{A}) = \ell\}$$

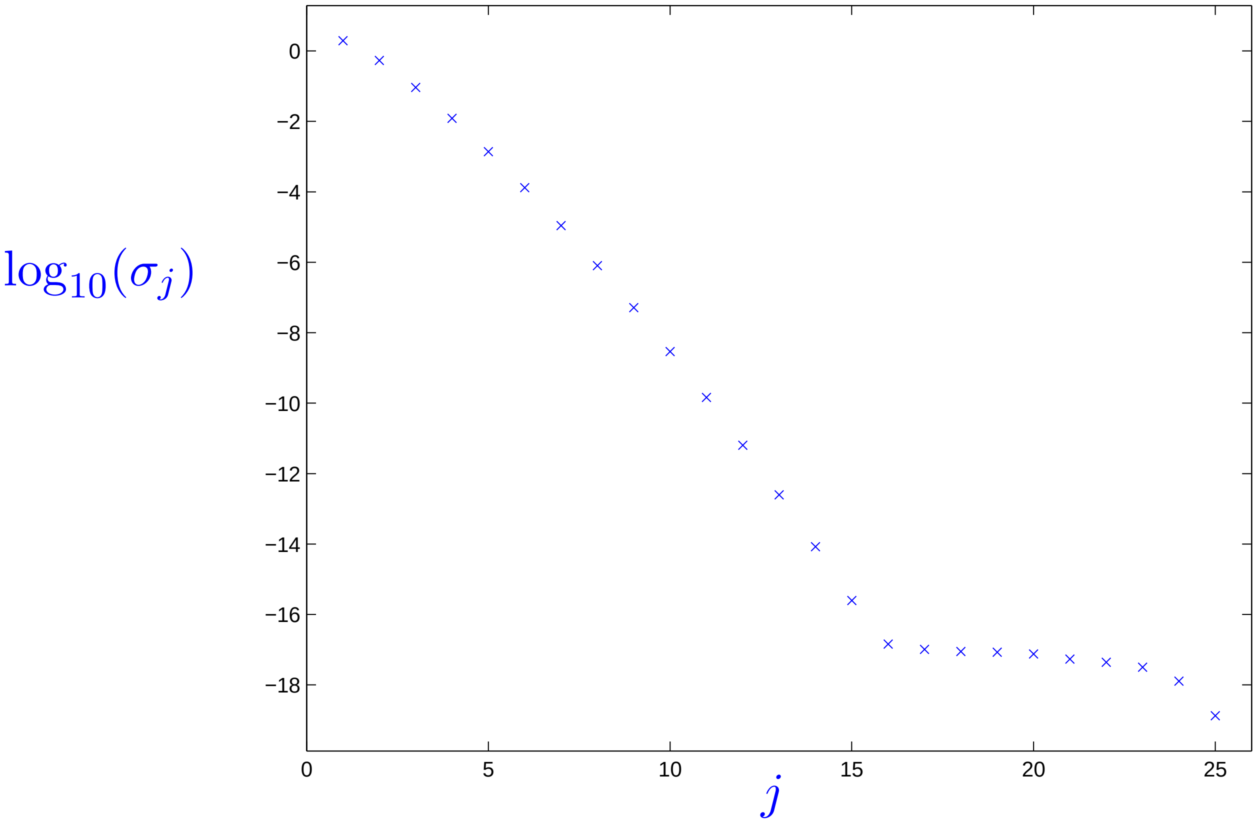
and

$$\text{argmin}\{||A - \tilde{A}|| :\ \text{rank}(\tilde{A}) = \ell\} = \sum_{j=1}^{\ell} \sigma_j\, u_j\, v_j^*.$$

Optimal solution: Set $Q_\ell = [u_1,\, u_2,\, \dots,\, u_\ell]$. Then $||A - Q_\ell\, Q_\ell^*\, A|| = \sigma_{\ell+1}$.

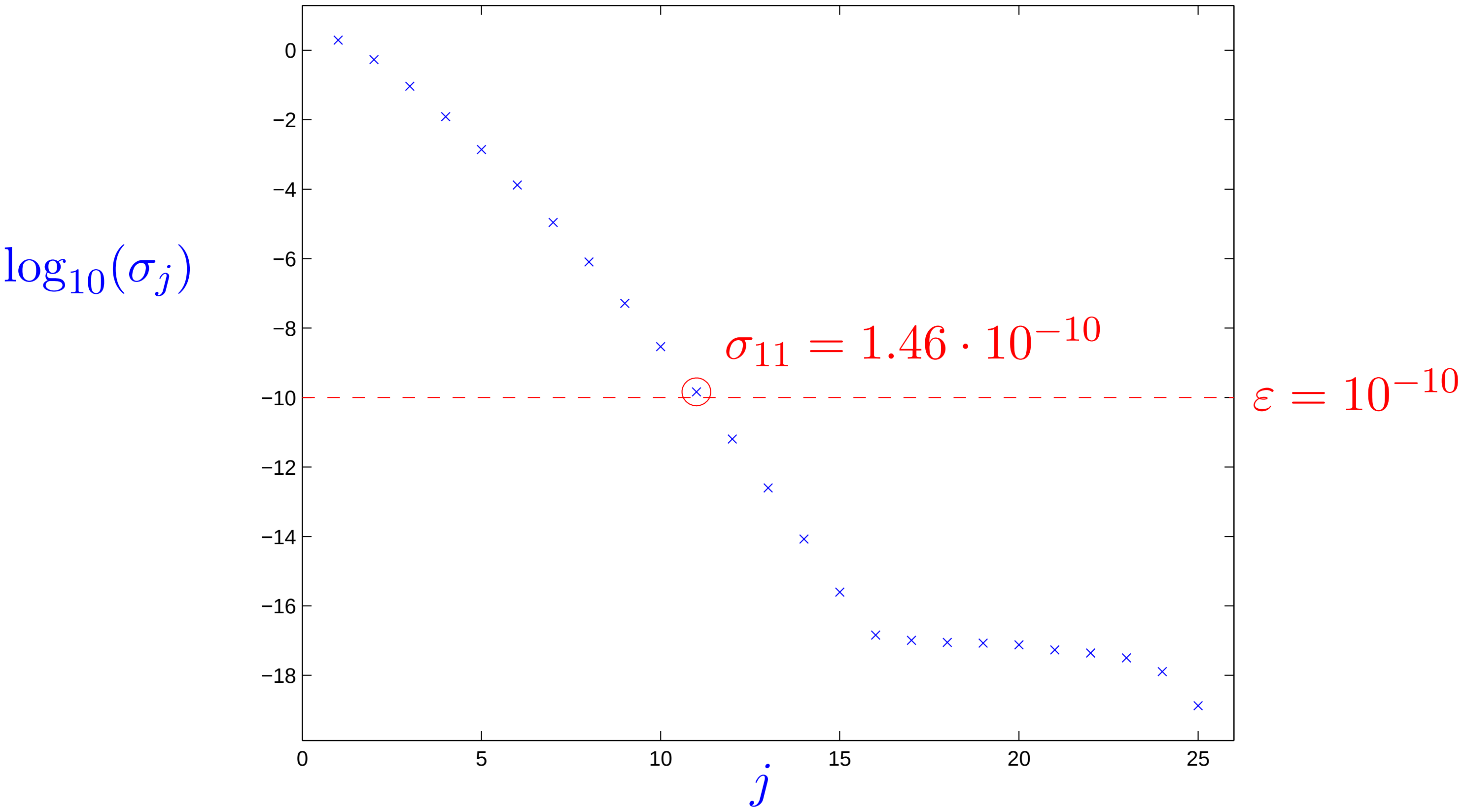
The decay of the singular values determines how well a matrix can be approximated by low-rank factorizations.

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.

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw random vectors $\omega_1,\, \omega_2,\, \omega_3,\, \dots \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later.)

2. Form "sample" vectors $y_1 = A\,\omega_1,\, y_2 = A\,\omega_2,\, y_3 = A\,\omega_3,\, \dots \in \mathbb{R}^m$.

3. Form orthonormal vectors $q_1,\, q_2,\, q_3,\, \dots \in \mathbb{R}^m$ such that
   $\text{Span}(q_1,\, q_2,\, \dots,\, q_\ell) = \text{Span}(y_1,\, y_2,\, \dots,\, y_\ell)$.
   (Using, for instance, Gram-Schmidt — no pivoting is required.)

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Solving the primitive problem via randomized sampling — intuition:

1. Draw random vectors $\omega_1,\, \omega_2,\, \omega_3,\, \cdots \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later.)

2. Form "sample" vectors $y_1 = A\,\omega_1,\, y_2 = A\,\omega_2,\, y_3 = A\,\omega_3,\, \cdots \in \mathbb{R}^m$.

3. Form orthonormal vectors $q_1,\, q_2,\, q_3,\, \cdots \in \mathbb{R}^m$ such that
   $\text{Span}(q_1,\, q_2,\, \dots,\, q_\ell) = \text{Span}(y_1,\, y_2,\, \dots,\, y_\ell)$.
   (Using, for instance, Gram-Schmidt — no pivoting is required.)

Surprise!
$\{q_j\}_{j=1}^{\ell}$ often does almost as good of a job as the singular vectors $\{u_j\}_{j=1}^{\ell}$!

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

Randomized algorithm — formal description:

1. Construct a random matrix Ω_ℓ of size $n \times \ell$.
   Suppose for now that Ω_ℓ is Gaussian.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\, Q_\ell^*\, Y_\ell$.
   (In other words, the columns of Q_ℓ form an ON basis for $\mathrm{Ran}(Y_\ell)$.)

Error measure:

The error incurred by the algorithm is $e_\ell = ||A - Q_\ell\, Q_\ell^*\, A||$.

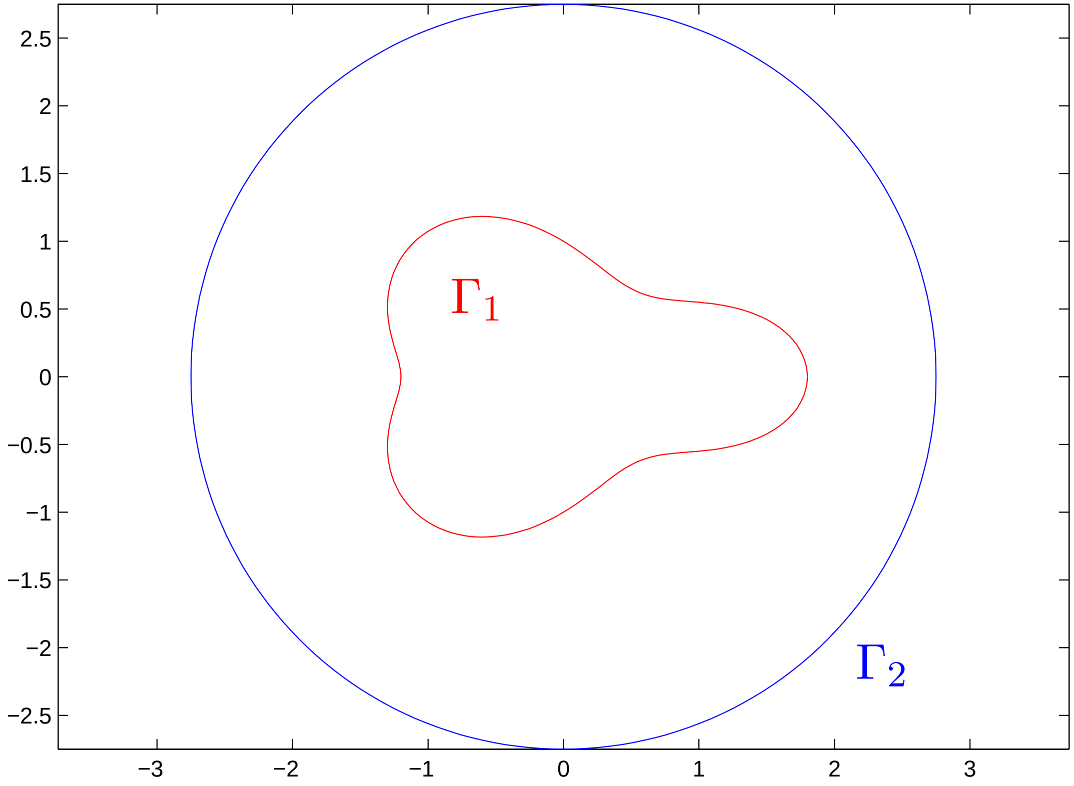
The error e_ℓ is bounded from below by $\sigma_{\ell+1} = \inf\{||A - B|| \,:\, B \text{ has rank } \ell\}$.

Specific example to illustrate the performance:

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

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

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



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

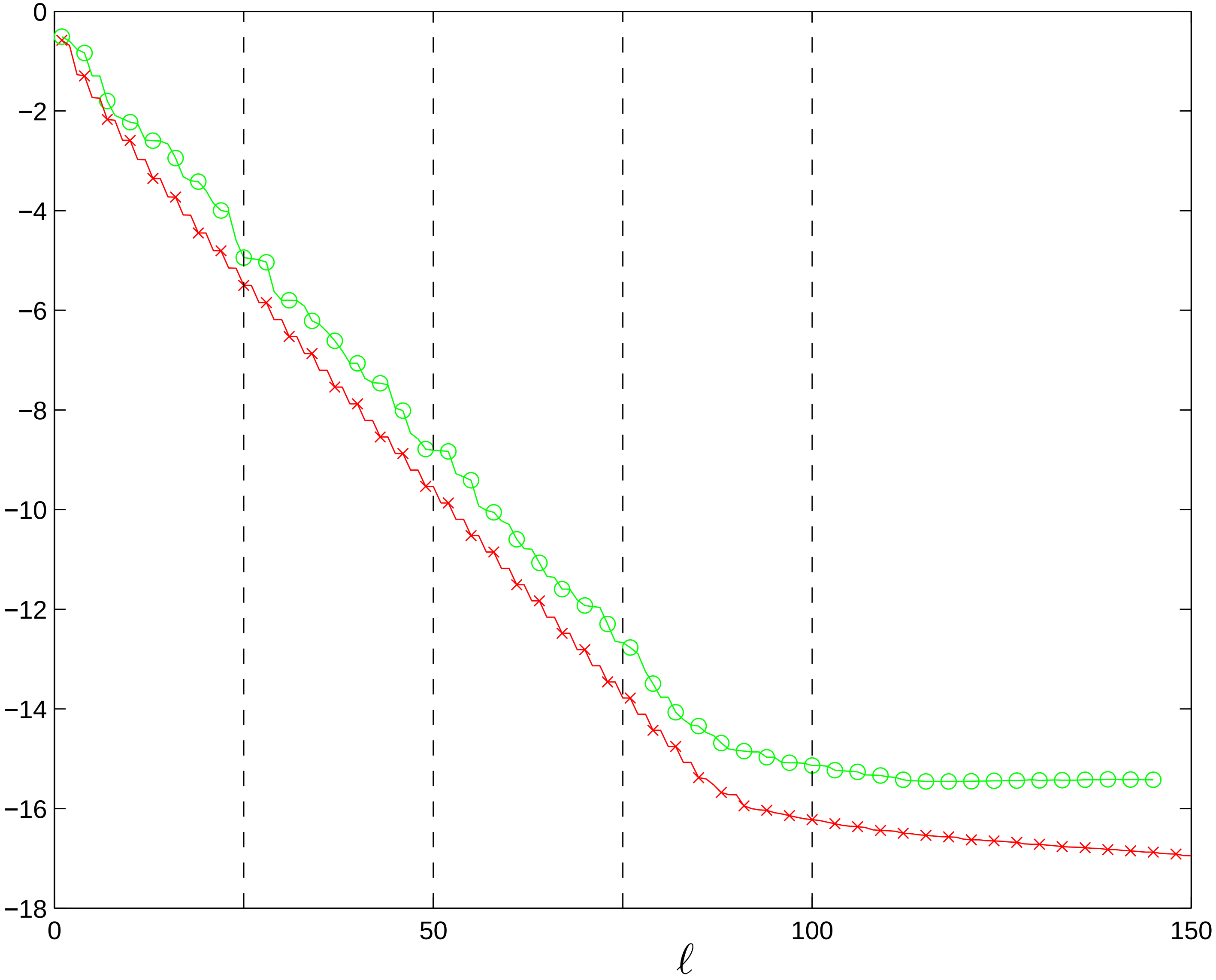
Results from one realization of the randomized algorithm

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

(actual error)

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

(theoretically minimal error)



Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

1. Construct a random matrix Ω_ℓ of size $n \times \ell$.
   Suppose for now that Ω_ℓ is Gaussian.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\, \Omega_\ell$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\, Q_\ell^*\, Y_\ell$.
   (In other words, the columns of Q_ℓ form an ON basis for $\text{Ran}(Y_\ell)$.)

Error measure:

The error incurred by the algorithm is $e_\ell = ||A - Q_\ell\, Q_\ell^*\, A||$.

The error e_ℓ is bounded from below by $\sigma_{\ell+1} = \inf\{||A - B|| :\ B \text{ has rank } \ell\}$.

Primitive problem: Given an $m \times n$ matrix A, and an integer ℓ, find an $m \times \ell$ orthonormal matrix Q_ℓ such that $A \approx Q_\ell\, Q_\ell^*\, A$.

1. Construct a random matrix Ω_ℓ of size $n \times \ell$.
   Suppose for now that Ω_ℓ is Gaussian.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\, Q_\ell^*\, Y_\ell$.
   (In other words, the columns of Q_ℓ form an ON basis for $\text{Ran}(Y_\ell)$.)

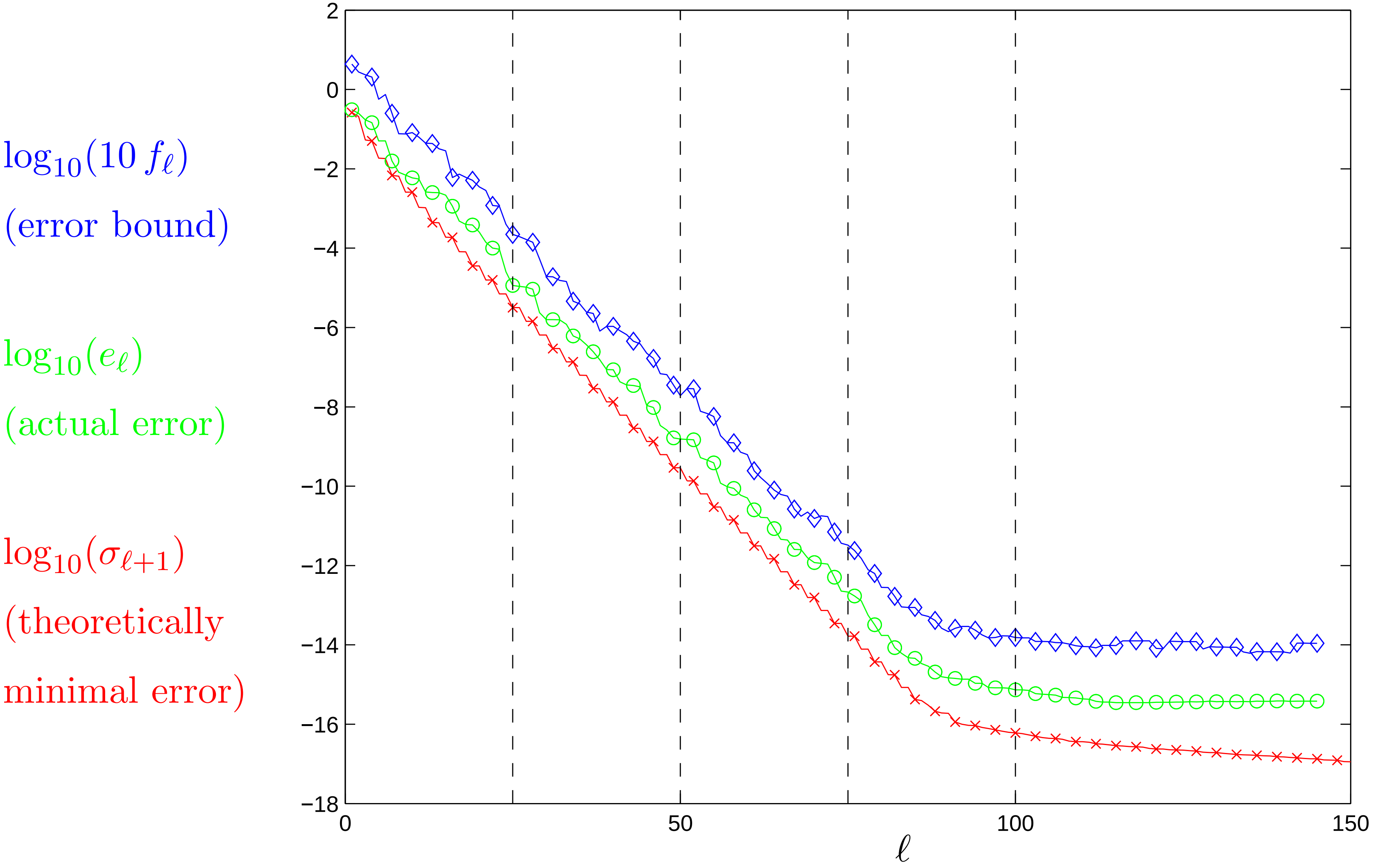
Error measure:

The error incurred by the algorithm is $e_\ell = ||A - Q_\ell\, Q_\ell^*\, A||$.

The error e_ℓ is bounded from below by $\sigma_{\ell+1} = \inf\{||A - B|| :\ B \text{ has rank } \ell\}$.

Error estimate: $f_\ell = \max_{1 \leq j \leq 10} \big|\big|\big(I - Q_\ell\, Q_\ell^*\big)\, y_{\ell+j}\big|\big|$.
The computation stops when we come to an ℓ such that $f_\ell < \varepsilon \times [\text{constant}]$.

Results from one realization of the randomized algorithm



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

(error bound)

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

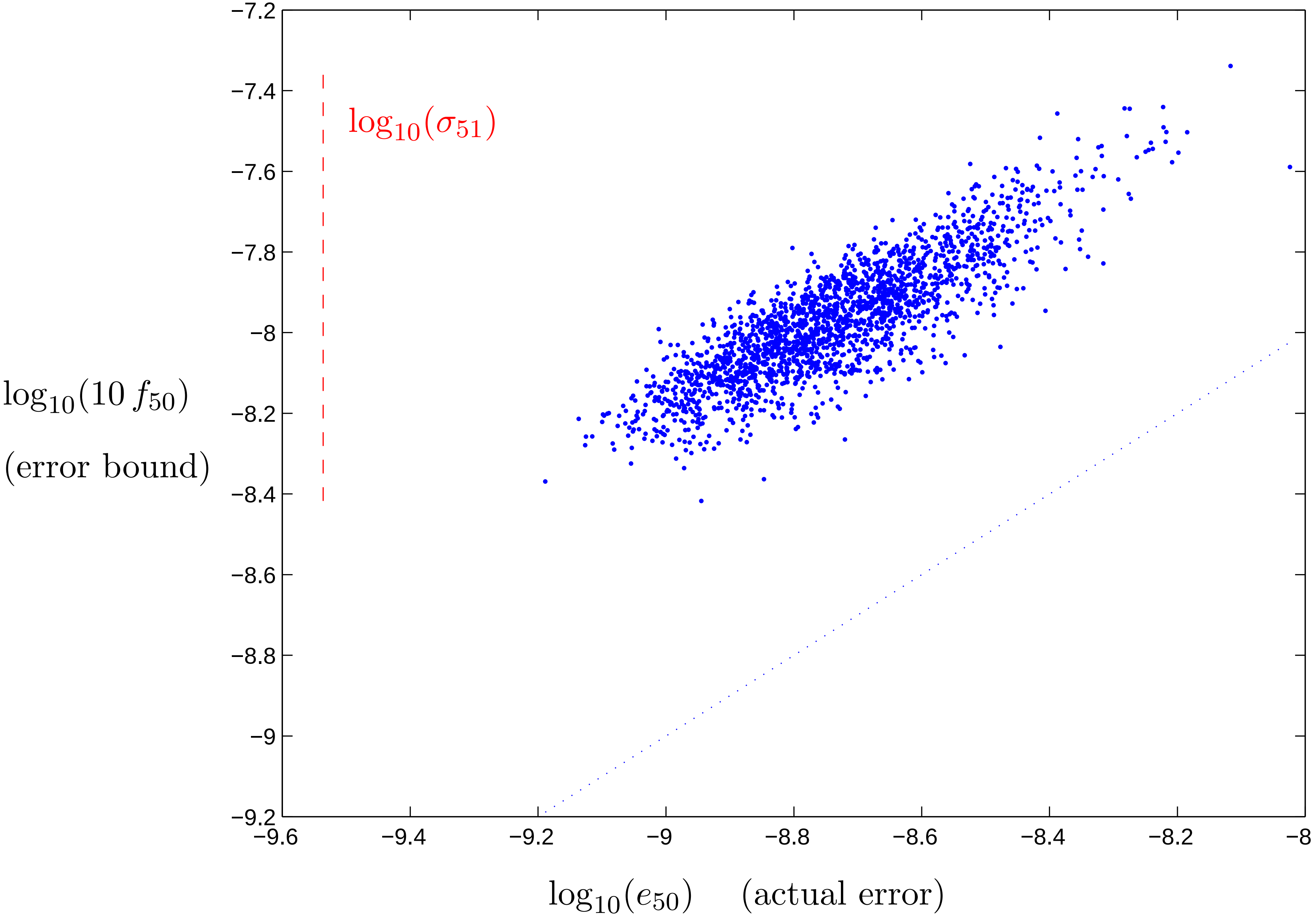
(actual error)

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

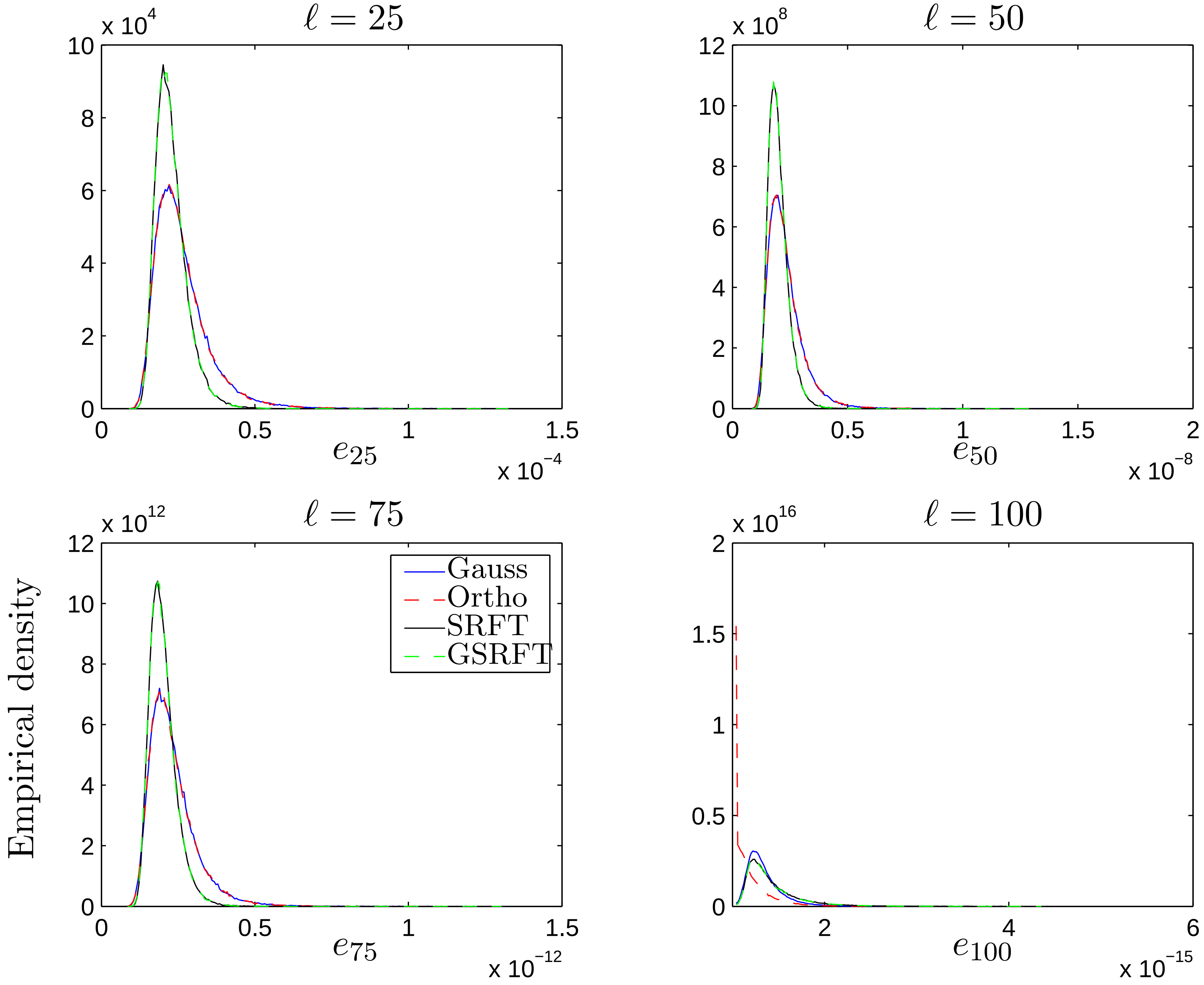
(theoretically minimal error)

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 $\ell = 50$ samples:



$\ell = 25$
x 10^4
10
8
6
4
2
0
0
0.5
1
1.5
e_{25}
x 10^-4
$\ell = 50$
x 10^8
12
10
8
6
4
2
0
0
0.5
1
1.5
2
e_{50}
x 10^-8
$\ell = 75$
x 10^12
12
10
8
6
4
2
0
0
0.5
1
1.5
e_{75}
x 10^-12
Gauss
Ortho
SRFT
GSRFT
Empirical density
$\ell = 100$
x 10^16
2
1.5
1
0.5
0
2
4
6
e_{100}
x 10^-15

Important:

- What is stochastic in practice 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)

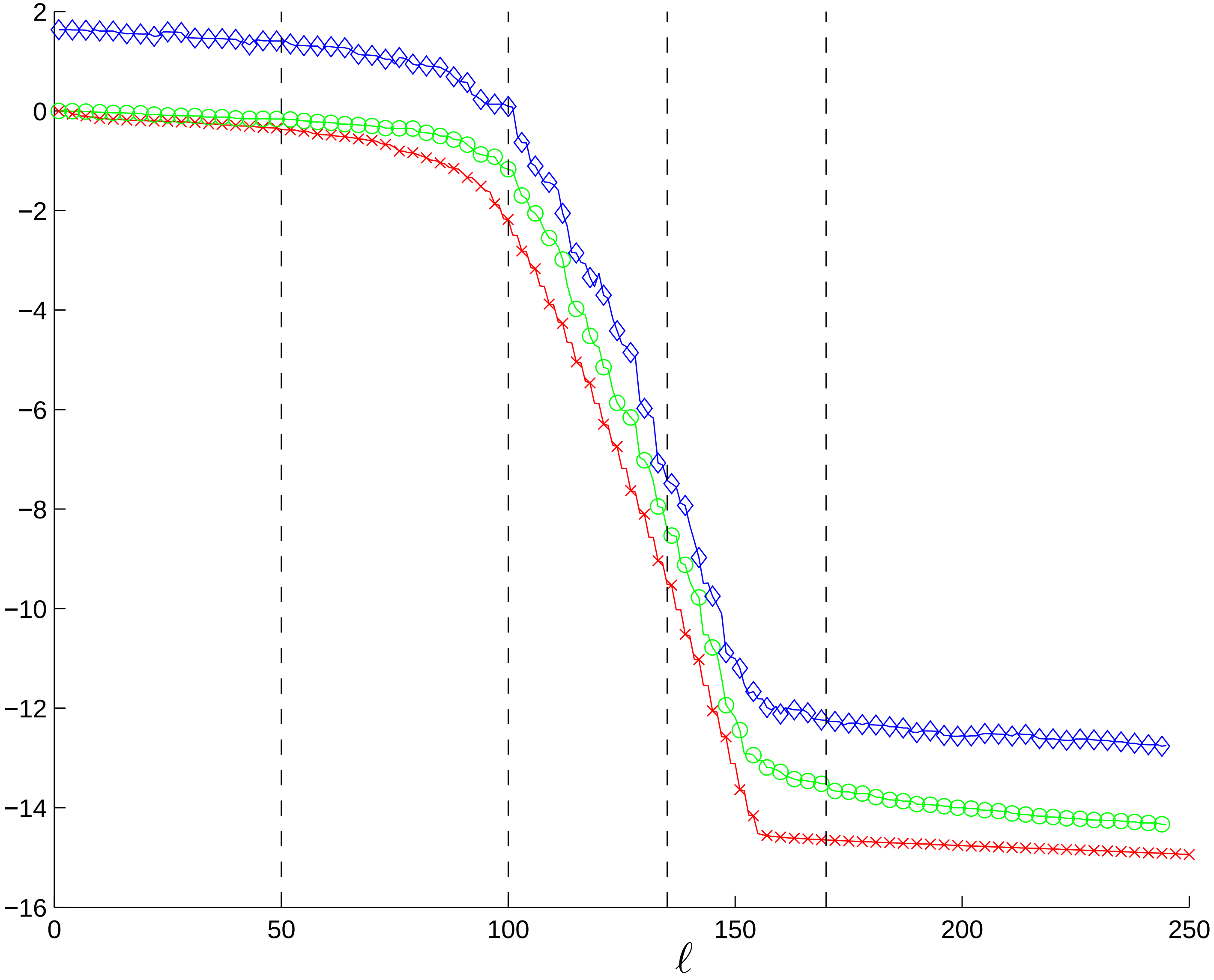
$\log_{10}(e_\ell)$

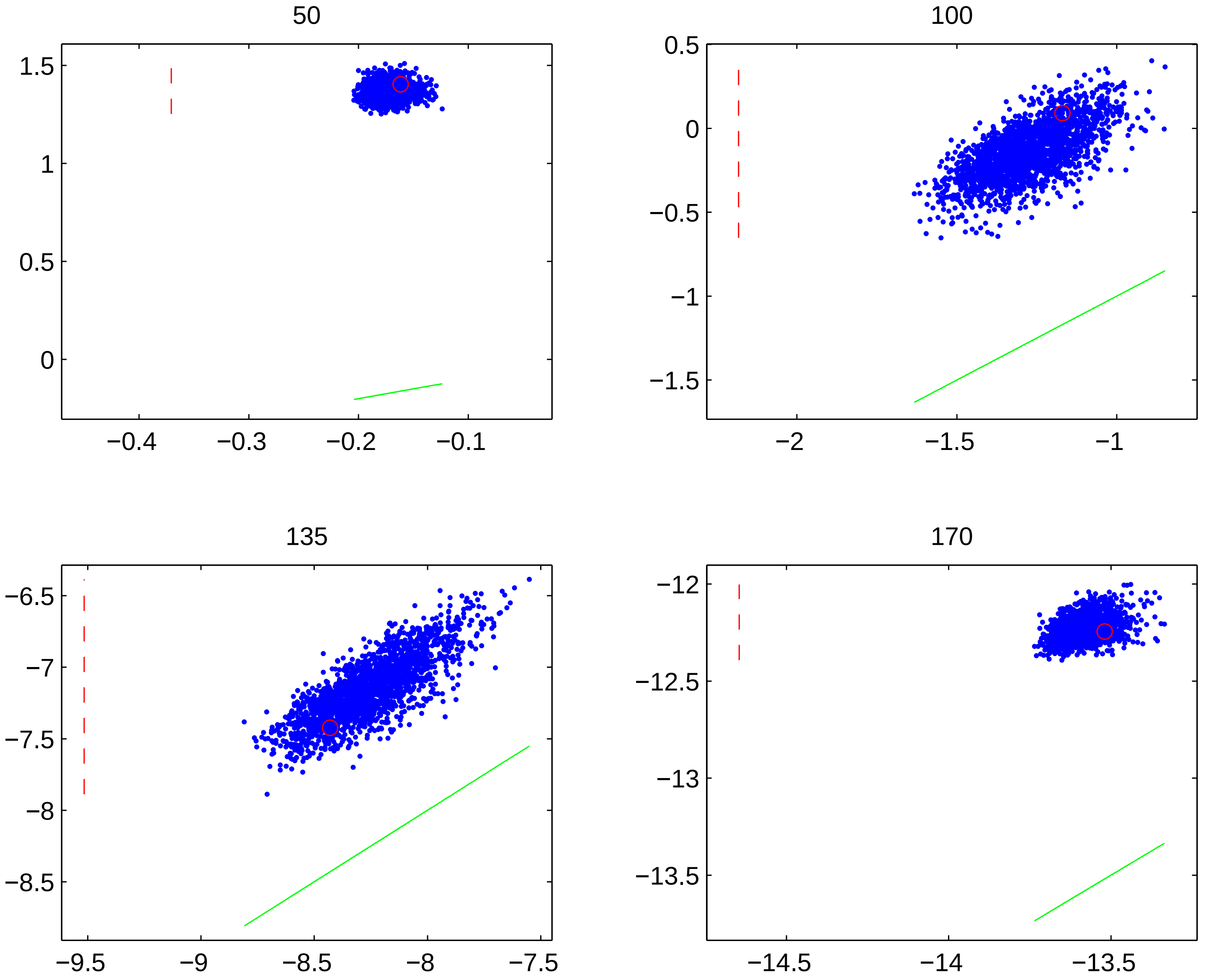
(actual error)

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

(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

What is the asymptotic complexity of the randomized method?

How does it compare to state-of-the-art non-randomized methods?

To answer such questions, we need to specify the computational environment; in what follows we consider three representative ones.

Sample Environment 1 (out of 3): The product $x \mapsto A\,x$ can be evaluated rapidly

Suppose that the cost of evaluating $x \mapsto A\,x$ is $O(m+n)$.
(Suppose A is sparse, FFT-able (there will be a log-factor), $\dots$)

1. Construct a Gaussian random matrix Ω_ℓ of size $n \times \ell$.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.
   Cost is $O(\ell\,(m+n))$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\,Q_\ell^*\,Y_\ell$.
   Cost is $O(\ell^2\,m)$.

The asymptotic cost is the same as that of Arnoldi/Lanzcos methods, but the randomized technique is more robust, and much better suited for implementation on parallel computers.

Sample Environment 2 (out of 3): No fast multiply, A fits in RAM

First let us try a Gaussian random matrix:

1. Construct a Gaussian random matrix Ω_ℓ of size $n \times \ell$.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.
   Cost is $O(\ell\,m\,n)$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\,Q_\ell^*\,Y_\ell$.
   Cost is $O(\ell^2\,m)$.

The asymptotic scaling is the same as rank-revealing QR!

Sample Environment 2 (out of 3):
No fast multiply, A fits in RAM

Let us change the random matrix Ω:

1. Construct an "SRFT" random matrix Ω_ℓ of size $n \times \ell$.
   An "SRFT" admits evaluation of $x \mapsto x\,\Omega$ in $O(n\,\log(\ell))$ operations.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.
   Cost is $O(m\,n\,\log(\ell))$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\,Q_\ell^*\,Y_\ell$.
   Cost is $O(m\,\ell^2)$.

Now the total cost is $O(m\,n\,\log(\ell))$.

We have $\ell \sim k$ so the cost is in fact $O(m\,n\,\log(k))$!

What is an “SRFT”?

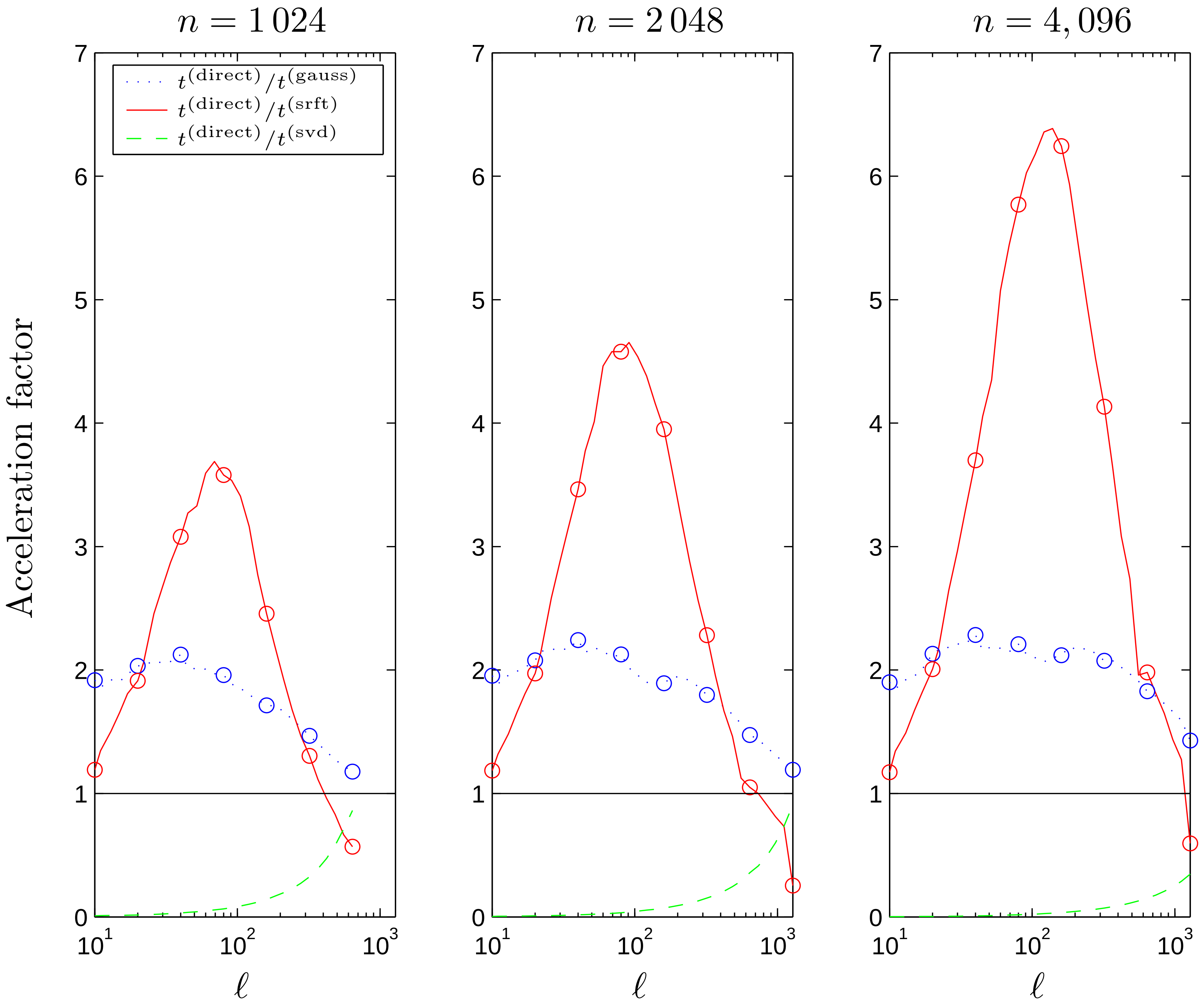
A Subsampled Random Fourier Transform. A random matrix with structure.

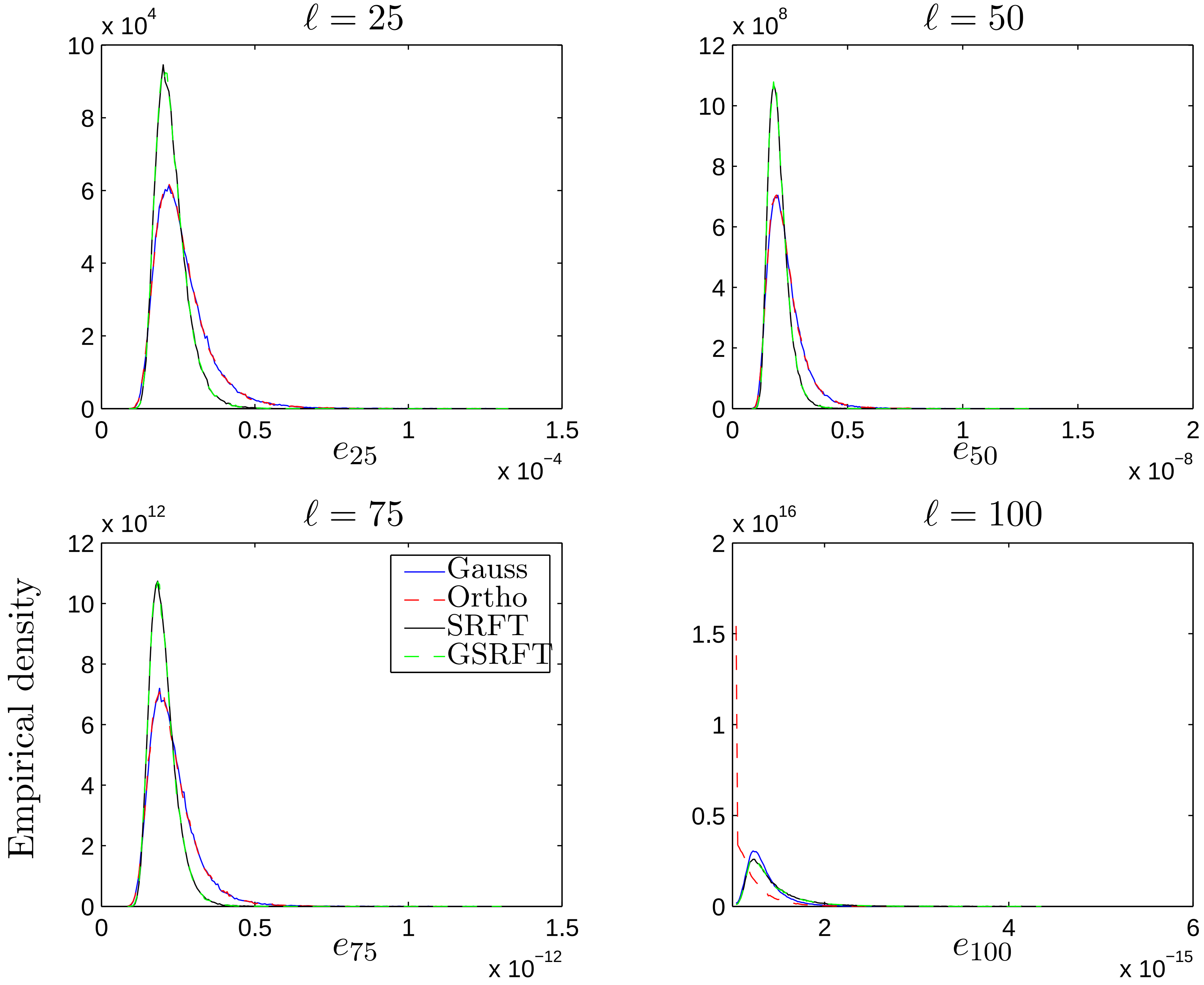
One possible choice:

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

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, $F_{jk} = \frac{1}{N^{1/2}}\,e^{-2\pi i(j-1)(k-1)/N}$.
- 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 $D\,F$.)


$n = 1\,024$
$n = 2\,048$
$n = 4,096$
Acceleration factor
$t^{(\mathrm{direct})}/t^{(\mathrm{gauss})}$
$t^{(\mathrm{direct})}/t^{(\mathrm{srft})}$
$t^{(\mathrm{direct})}/t^{(\mathrm{svd})}$
0
1
2
3
4
5
6
7
10^1
10^2
10^3
ℓ



$\ell = 25$
x 10^4
10
8
6
4
2
0
0
0.5
1
1.5
e_{25}
x 10^-4
$\ell = 50$
x 10^8
12
10
8
6
4
2
0
0
0.5
1
1.5
2
e_{50}
x 10^-8
$\ell = 75$
x 10^12
12
10
8
6
4
2
0
0
0.5
1
1.5
e_{75}
x 10^-12
Empirical density
Gauss
Ortho
SRFT
GSRFT
$\ell = 100$
x 10^16
2
1.5
1
0.5
0
2
4
6
e_{100}
x 10^-15

Notes:

- Significant speed-ups are achieved for common problem sizes. For instance, $m = n = 2\,000$ and $k = 200$ leads to a speed-up by roughly a factor of 4.

- Many other choices of random matrices have been found.
  - Subsampled Hadamard transform.
  - Wavelets.
  - Random chains of Given's rotations. (Seems to work the best.)

- Theory is poorly understood. Vastly overstates the risk of inaccurate results.

- This idea was proposed by Liberty, Rokhlin, Tygert, Woolfe (2006).
  - The SRFT described above was also suggested by Nir Ailon and Bernard Chazelle (2006) in a related context.
  - Related recent work by Sarlós (on randomized regression).
  - Very interesting follow-up paper on overdetermined linear least-squares regression by Rokhlin and Tygert (2008).

Sample Environment 3 (out of 3):

No fast multiply, A is huge and must be stored on disk.

1. Construct a Gaussian random matrix Ω_ℓ of size $n \times \ell$.

2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.

   Cost is $O(\ell\, m\, n)$ flops and one pass over the matrix.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\, Q_\ell^*\, Y_\ell$.

   Cost is $O(\ell^2\, m)$ flops.

Asymptotic flop counts are very similar to those of classical methods.

Sample Environment 3 (out of 3): No fast multiply, A is huge and must be stored on disk.

1. Construct a Gaussian random matrix Ω_ℓ of size $n \times \ell$.
2. Form the $m \times \ell$ sample matrix $Y_\ell = A\,\Omega_\ell$.
   Cost is $O(\ell\, m\, n)$ flops and one pass over the matrix.
3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $Y_\ell = Q_\ell\, Q_\ell^*\, Y_\ell$.
   Cost is $O(\ell^2\, m)$ flops.

Asymptotic flop counts are very similar to those of classical methods.

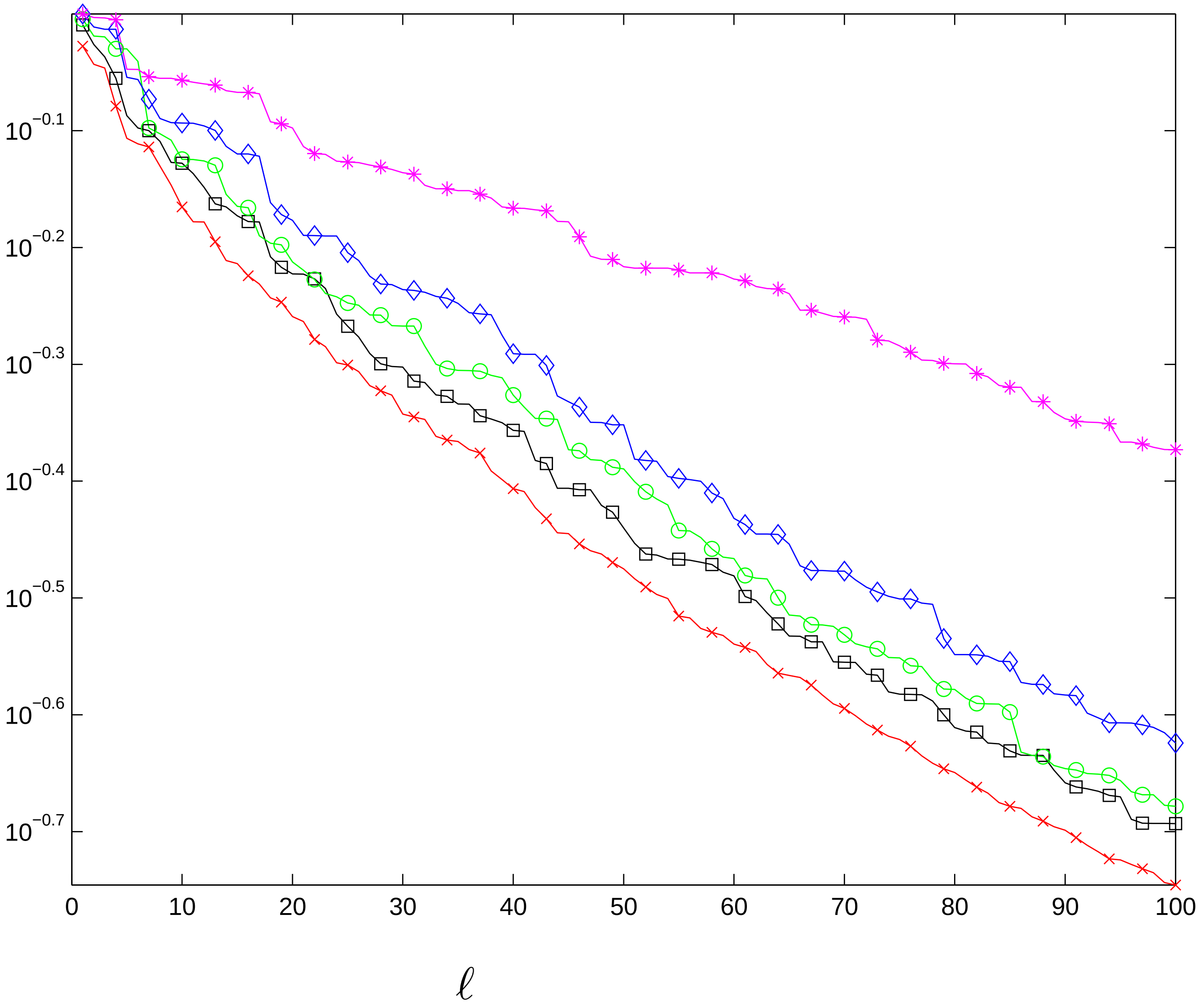
But flop count is largely irrelevant here — the cost of arithmetic is dwarfed by the memory access time, which is minimal for the randomized algorithm.

Similar considerations apply to implementation on multi-core architectures, streaming data, *etc.*

Problem: Poor error control!

10-log of e_ℓ

is shown in pink



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.

Theory

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)

There have been significant progress in understanding and minimizing the error over the last few years.

The framework in the following theorems is:

We are given an $m \times n$ matrix A and seek a rank-k approximation

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

Fix a small integer p representing how much we "over-sample". Set $\ell = k + p$.

1. Construct a Gaussian random matrix Ω_ℓ of size $n \times \ell$.
2. Form the $m \times \ell$ matrix $Y_\ell = A\,\Omega_\ell$.
3. Construct an $m \times \ell$ orthogonal matrix Q_ℓ such that $Y_\ell = Q_\ell\,Q_\ell^*\,Y_\ell$.

Question: How does the error $||A - Q_\ell\,Q_\ell^*\,A||$ compare to σ_{k+1}?

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 $\quad \sigma_{k+1} = \inf\{||A - B|| \,:\, \mathrm{rank}(B) = k\}.$

Then

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

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 $\quad \sigma_{k+1} = \inf\{||A - B|| \,:\, \mathrm{rank}(B) = k\}.$

Then

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

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:

$$||A - Q\,Q^*\,A||_2 \leq 10\;\sqrt{(k+p)\,(n-k)}\;\sigma_{k+1}.$$

The factor in red 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 $\sigma_1, \sigma_2, \sigma_3, \dots$. Choose integers $k \geq 1$ and $p \geq 2$, and draw an $n \times (k+p)$ standard Gaussian random matrix Ω. Construct the sample matrix $Y = A\Omega$, and let Q denote an orthonormal matrix such that $Ran(Q) = Ran(Y)$. Then*

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

Moreover,

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

- Numerical experiments indicate that these estimates are close to sharp.

- When $\sigma_j \sim \beta^j$, we have $\left(\sum_{j=k+1}^{\infty} \sigma_j^2\right)^{1/2} \sim \sigma_{k+1} \frac{1}{\sqrt{1-\beta^2}}$.

Due to overwhelmingly strong concentration of mass effects, tail probabilities are often in a practical sense irrelevant.

Due to overwhelmingly strong concentration of mass effects, tail probabilities are often in a practical sense irrelevant. However:

Theorem: *[Halko, Martinsson, Tropp 2009] Fix a* real *$m\times n$ matrix A with singular values $\sigma_1, \sigma_2, \sigma_3, \dots$. Choose integers $k \geq 1$ and $p \geq 4$, and draw an $n \times (k+p)$ standard Gaussian random matrix Ω. Construct the sample matrix $Y = A\,\Omega$, and let Q denote an orthonormal matrix such that $Ran(Q) = Ran(Y)$. For all $u, t \geq 1$,*

$$||A - QQ^{*}A|| \leq \left(1 + t\,\sqrt{12\,k/p} + u\,t\,\frac{e\,\sqrt{k+p}}{p+1}\right)\,\sigma_{k+1} + \frac{t\,e\,\sqrt{k+p}}{p+1}\,\left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

except with probability at most $5\,t^{-p} + 2\,e^{-u^2/2}$.

The theorem can be simplified by by choosing t and u appropriately. For instance,

$$||A - QQ^{*}A|| \leq \left(1 + 8\sqrt{(k+p)\cdot p\log p}\right)\,\sigma_{k+1} + 3\sqrt{k+p}\,\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2},$$

except with probability at most $6\,p^{-p}$.

Power method for improving accuracy:

[Rokhlin, Tygert, Szlam 2008]

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

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

Idea: The matrix $B = (A\,A^{*})^{q}\,A$ 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:

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 $Z = B\,\Omega$. Let P_Z denote the orthogonal projector onto the range of Z. Then*

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

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

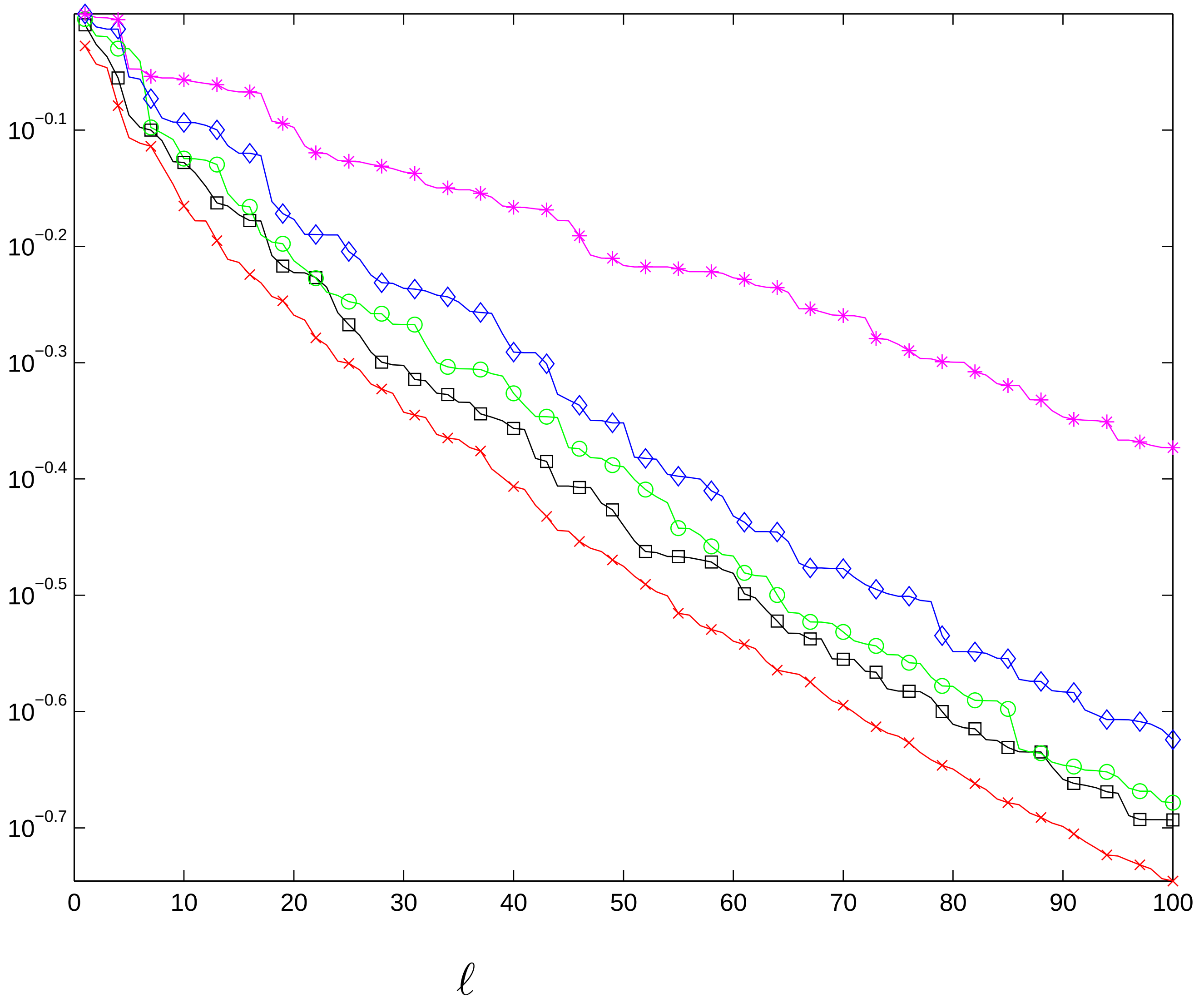
the power method with:

$q = 0$ 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.

Example: Eigenfaces

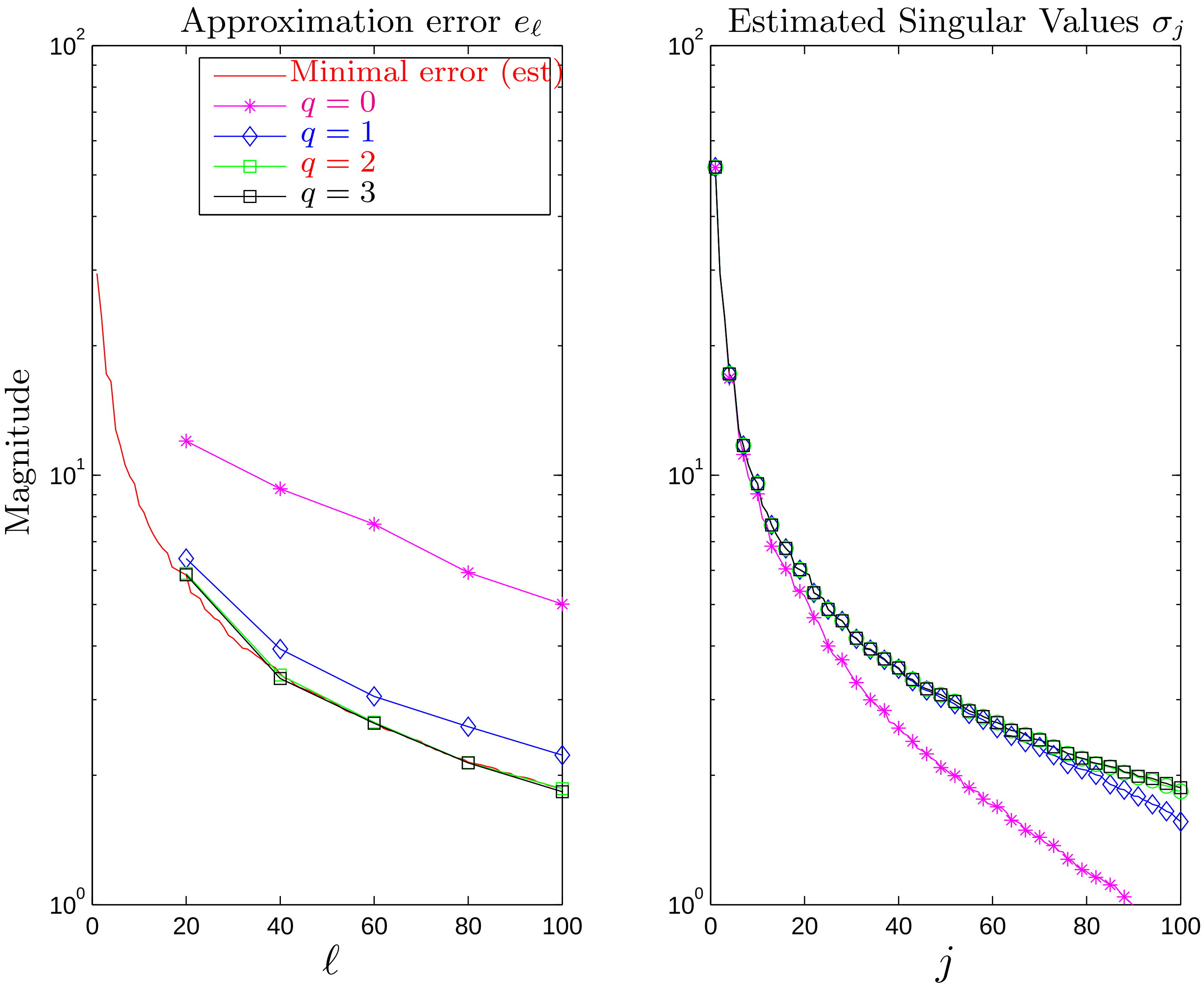
We next process process a data base containing $m = 7\,254$ pictures of faces

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 A.

The left singular vectors of A are the so called *eigenfaces* of the data base.

We compute these using the randomized method. This method requires only $2q + 1$ passes over the data, where q is the parameter in the power scheme (we evaluate the singular vectors of $(A\,A^*)^q\,A$).


Approximation error e_ℓ
Magnitude
10^2
10^1
10^0
0
20
40
60
80
100
ℓ
Minimal error (est)
$q = 0$
$q = 1$
$q = 2$
$q = 3$
Estimated Singular Values σ_j
10^2
10^1
10^0
0
20
40
60
80
100
j

In more recent work, we demonstrated the capabilities of the “low pass count” method by processing a $200\,000 \times 200\,000$ matrix stored on disk.

The matrix has about 200 significant singular modes; the remaining singular values level out at about 5% of the magnitude of the dominant singular value.

The randomized method accurately constructs the first 200 singular values in a couple of hours on a standard desktop PC.

[Forthcoming paper with Mark Tygert and Yoel Shkolnisky.]

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*

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

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

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

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, (1), this map satisfies (2) 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 LAPACK. Acceleration of the Fast Multipole Method.

- Application to physical operators: model reduction / coarse graining / . . .

- New estimates on spectral properties of random matrices.

- 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 / . . .

Reference: *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions* N. Halko, P.G. Martinsson, J. Tropp. Sep. 2009.

Reference:

Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions N. Halko, P.G. Martinsson, J. Tropp. Sep. 2009.

Published as arXiv.org report 0909.4061. In review..