Enabling very large-scale matrix computations via randomization

Gunnar Martinsson
The University of Colorado at Boulder

Students:

Adrianna Gillman

Nathan Halko

Patrick Young

Collaborators:

Edo Liberty (Yale)

Vladimir Rokhlin (Yale)

Yoel Shkolnisky (Tel Aviv University)

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

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

When $k \ll \min(m, n)$, computing the factors B and C is advantageous:

- Storing B and C requires $(m+n)\,k$ units of storage instead of $m\,n$ units.
- A matrix-vector multiply requires $(m+n)\,k$ flops instead of $m\,n$ flops.
- Certain factorizations reveal properties of the matrix → PCA

Excellent algorithms exist. Most were designed for matrices that fit in RAM, and are not suitable for emerging applications where *communication* is the bottleneck:

- *Multi-processor computing.* CPU speed is growing slowly, but processors get cheaper all the time. Inter-processor communication is expensive.
- *Very large data sets.* A result of cheap (slow) storage and cheap information gathering devices (*e.g.* automatic DNA sequencing, sensor networks, . . .)
- *Streaming data.*

Real life complications: We are rarely lucky enough to work with matrices of exact rank k. Instead, we have to consider approximation problems:

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

$$\min\{||\mathsf{A} - \mathsf{B}|| : \ \mathrm{rank}(\mathsf{B}) = k\} \leq \varepsilon.$$

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

$$\mathsf{A}_{(k)} = \mathrm{argmin}\{||\mathsf{A} - \mathsf{B}|| : \ \mathrm{rank}(\mathsf{B}) = k\}.$$

Moreover, there might be constraints on the factors:

Problem (SVD): Given A and a precision ε, find a minimal k, and orthonormal rank-k matrices $\mathsf{U}_{(k)}$ and $V_{(k)}$, and a diagonal matrix $\Sigma_{(k)}$ such that

$$||\mathsf{A} - \mathsf{U}_{(k)}\,\Sigma_{(k)}\,V_{(k)}^{*}|| \leq \varepsilon.$$

Problem (CUR): Given A and a precision ε, find a minimal k, and matrices $\mathsf{C}_{(k)}$ and $\mathsf{R}_{(k)}$ consisting of k columns and k rows of A such that

$$||\mathsf{A} - \mathsf{C}_{(k)}\,\mathsf{U}_{(k)}\,\mathsf{R}_{(k)}|| \leq \varepsilon.$$

Prior work on related problems 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, *etc*)
- A. Deshpande and S. Vempala (2006)
- S. Friedland, M. Kaveh, A. Niknejad, and H. Zare (2006)
- T. Sarlós (2006a, 2006b, 2006c)

There are also strong connections between part of the described work and earlier work on so called *block Lanczos methods* [Cullman and Donath 1974], [Golub and Underwood 1977], [Grimes, Lewis, and Simon 1994] with random starting vectors.

Literature survey in: *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions* N. Halko, P.G. Martinsson, J. Tropp.

Did we overlook relevant work in the manuscript? Feedback is warmly welcomed!

The talk describes randomized algorithms for a *primitive problem* that we argue lies at the root of a broad range of approximation problems.

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

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 $\ell < \min(m, n)$, find an orthonormal $m \times \ell$ matrix Q such that $\mathsf{A} \approx \mathsf{Q}\mathsf{Q}^*\mathsf{A}$.

Any standard factorization can easily be obtained from Q*.*

To illustrate, suppose that we seek an approximate rank-ℓ SVD

$$\begin{array}{ccccc} \mathsf{A} & \approx & \mathsf{U} & \Sigma & \mathsf{V}^*, \\ m \times n & & m \times \ell & \ell \times \ell & \ell \times n \end{array}$$

where U and V are orthonormal, and Σ is diagonal with non-negative entries.

The following steps will do the job:

1. Form the (small) matrix $\mathsf{B} = \mathsf{Q}^*\mathsf{A}$.
2. Compute the SVD of the (small) matrix $\mathsf{B} = \hat{\mathsf{U}}\Sigma\mathsf{V}^*$.
3. Set $\mathsf{U} = \mathsf{Q}\hat{\mathsf{U}}$.

Note: The *Golub-Businger algorithm* is very similar. In GS, you solve the "primitive problem" via QR. This directly yields a factorization $\mathsf{A} \approx \mathsf{QRP}$; then simply set $\mathsf{B} = \mathsf{RP}$ in Step 2, and execute Step 3 as described.

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

Solving the primitive problem via randomized sampling — intuition:

1. Draw random vectors $\boldsymbol{\omega}_1,\, \boldsymbol{\omega}_2,\, \boldsymbol{\omega}_3,\, \cdots \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later — think Gaussian for now.)

2. Form “sample” vectors $\boldsymbol{y}_1 = \mathsf{A}\,\boldsymbol{\omega}_1,\, \boldsymbol{y}_2 = \mathsf{A}\,\boldsymbol{\omega}_2,\, \boldsymbol{y}_3 = \mathsf{A}\,\boldsymbol{\omega}_3,\, \cdots \in \mathbb{R}^m$.

3. Form orthonormal vectors $\boldsymbol{q}_1,\, \boldsymbol{q}_2,\, \boldsymbol{q}_3,\, \cdots \in \mathbb{R}^m$ such that

   $$\text{Span}(\boldsymbol{q}_1,\, \boldsymbol{q}_2,\, \dots,\, \boldsymbol{q}_\ell) = \text{Span}(\boldsymbol{y}_1,\, \boldsymbol{y}_2,\, \dots,\, \boldsymbol{y}_\ell).$$

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

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

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

Solving the primitive problem via randomized sampling — intuition:

1. Draw random vectors $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_3, \dots \in \mathbb{R}^n$.
   (We will discuss the choice of distribution later — think Gaussian for now.)

2. Form "sample" vectors $\boldsymbol{y}_1 = \mathsf{A}\,\boldsymbol{\omega}_1,\ \boldsymbol{y}_2 = \mathsf{A}\,\boldsymbol{\omega}_2,\ \boldsymbol{y}_3 = \mathsf{A}\,\boldsymbol{\omega}_3, \dots \in \mathbb{R}^m$.

3. Form orthonormal vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3, \dots \in \mathbb{R}^m$ such that
   $$\mathrm{Span}(\boldsymbol{q}_1,\ \boldsymbol{q}_2,\ \dots,\ \boldsymbol{q}_\ell) = \mathrm{Span}(\boldsymbol{y}_1,\ \boldsymbol{y}_2,\ \dots,\ \boldsymbol{y}_\ell).$$
   For instance, Gram-Schmidt can be used — pivoting is rarely required.

If A has *exact* rank ℓ, then $\mathrm{Span}\{\boldsymbol{q}_j\}_{j=1}^{\ell} = \mathrm{Ran}(\mathsf{A})$ with probability 1.

What is perhaps surprising is that even in the general case, $\{\boldsymbol{q}_j\}_{j=1}^{\ell}$ often does almost as good of a job as the theoretically optimal vectors (which happen to be the ℓ leading left singular vectors).

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

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 $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.

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

Error measure:

The error incurred by the algorithm is $e_\ell = ||\mathsf{A} - \mathsf{Q}_\ell\,\mathsf{Q}_\ell^{*}\,\mathsf{A}||$.

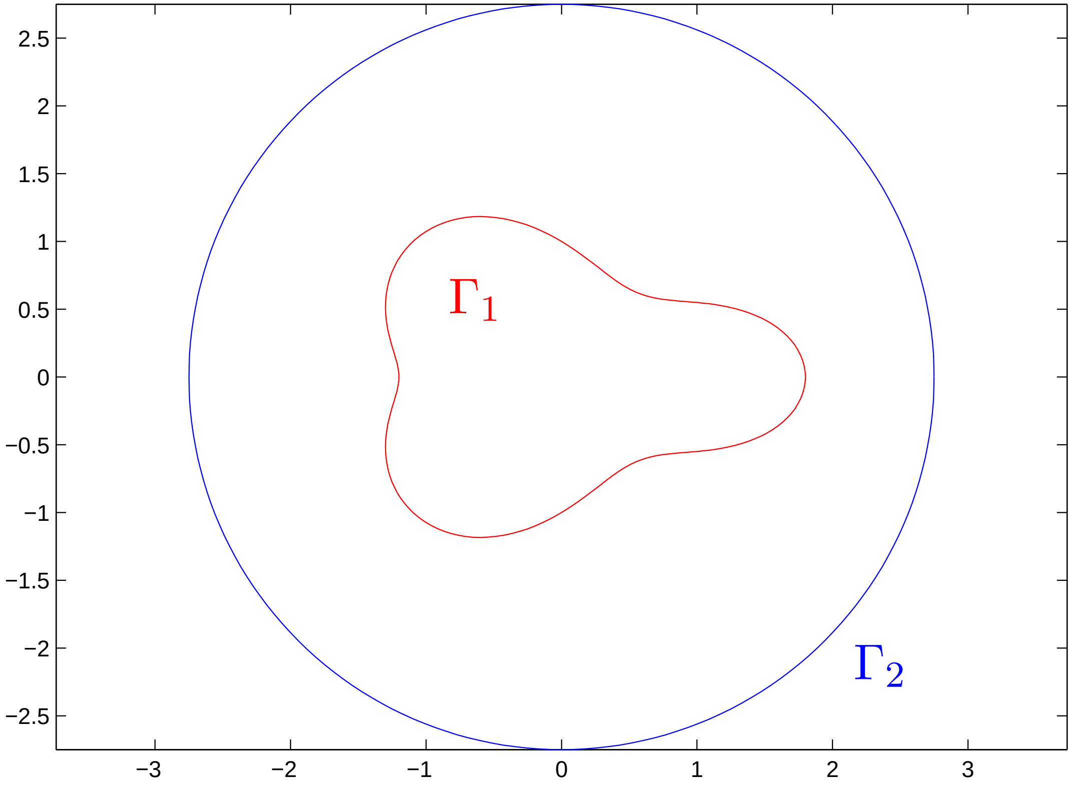
The error e_ℓ is bounded from below by $\sigma_{\ell+1} = \inf\{||\mathsf{A} - \mathsf{B}|| \, : \, \mathsf{B} \text{ has rank } \ell\}$.

Specific example to illustrate the performance:

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

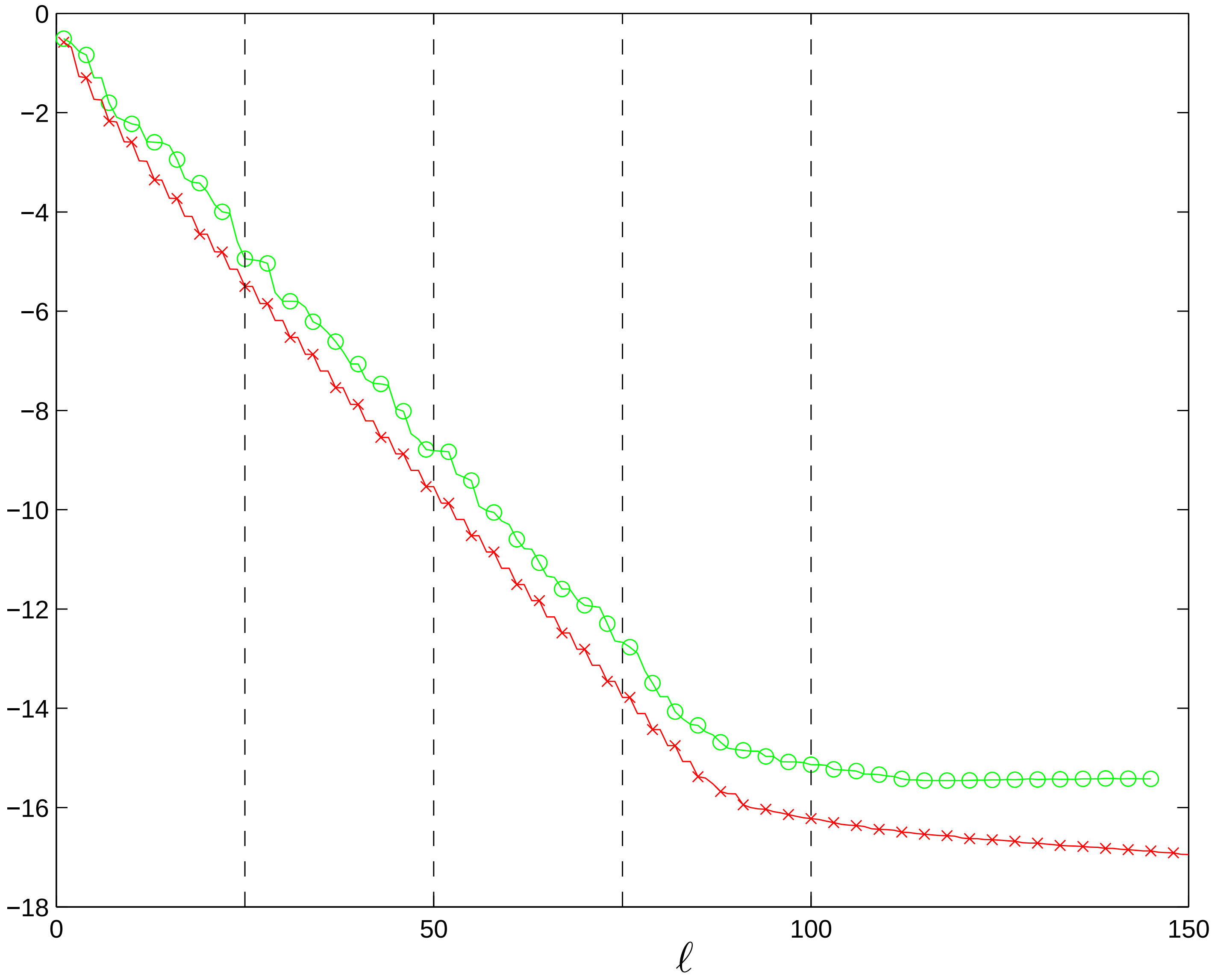
$$[\mathcal{S}_{\Gamma_2 \leftarrow \Gamma_1} u](\boldsymbol{x}) = \alpha \int_{\Gamma_1} \log|\boldsymbol{x} - \boldsymbol{y}|\, u(\boldsymbol{y})\, ds(\boldsymbol{y}), \qquad \boldsymbol{x} \in \Gamma_2,$$

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



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

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 $\mathsf{A} \approx \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{A}$.

1. Construct a random matrix $\mathsf{\Omega}_\ell$ of size $n \times \ell$.

   Suppose for now that $\mathsf{\Omega}_\ell$ is Gaussian.

2. Form the $m \times \ell$ sample matrix $\mathsf{Y}_\ell = \mathsf{A}\,\mathsf{\Omega}_\ell$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $\mathsf{Y}_\ell = \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{Y}_\ell$.

   (In other words, the columns of Q_ℓ form an ON basis for $\mathrm{Ran}(\mathsf{Y}_\ell)$.)

Error measure:

The error incurred by the algorithm is $e_\ell = ||\mathsf{A} - \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{A}||$.

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

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

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 $\mathsf{Y}_\ell = \mathsf{A}\, \Omega_\ell$.

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

Error measure:

The error incurred by the algorithm is $e_\ell = ||\mathsf{A} - \mathsf{Q}_\ell\, \mathsf{Q}_\ell^*\, \mathsf{A}||$.

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

Error estimate: $f_\ell = \max_{1 \leq j \leq 10} \left|\left|\left(\mathsf{I} - \mathsf{Q}_\ell\, \mathsf{Q}_\ell^*\right) y_{\ell+j}\right|\right|$.

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

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

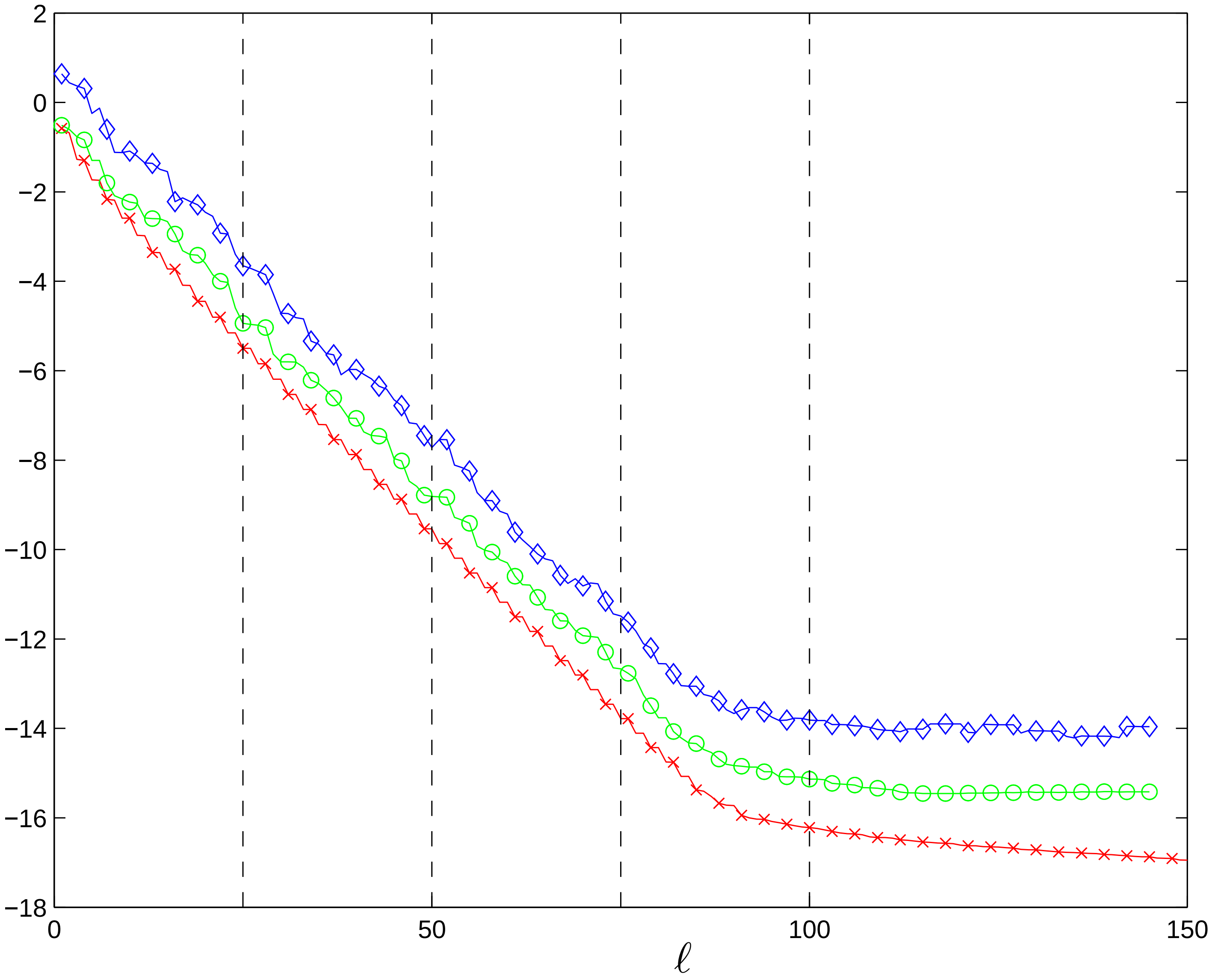
(error bound)

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

(actual error)

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

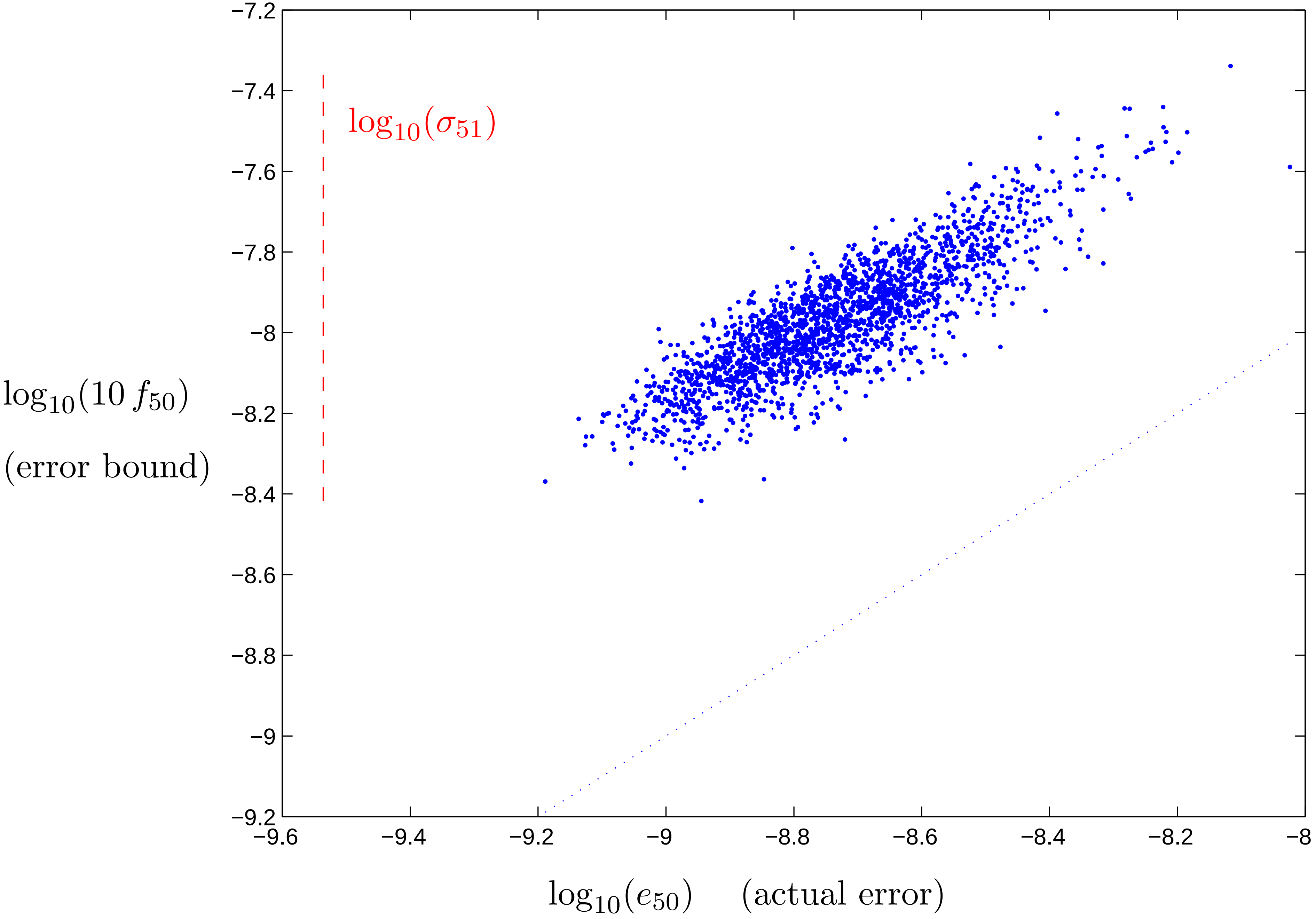
(theoretically minimal error)

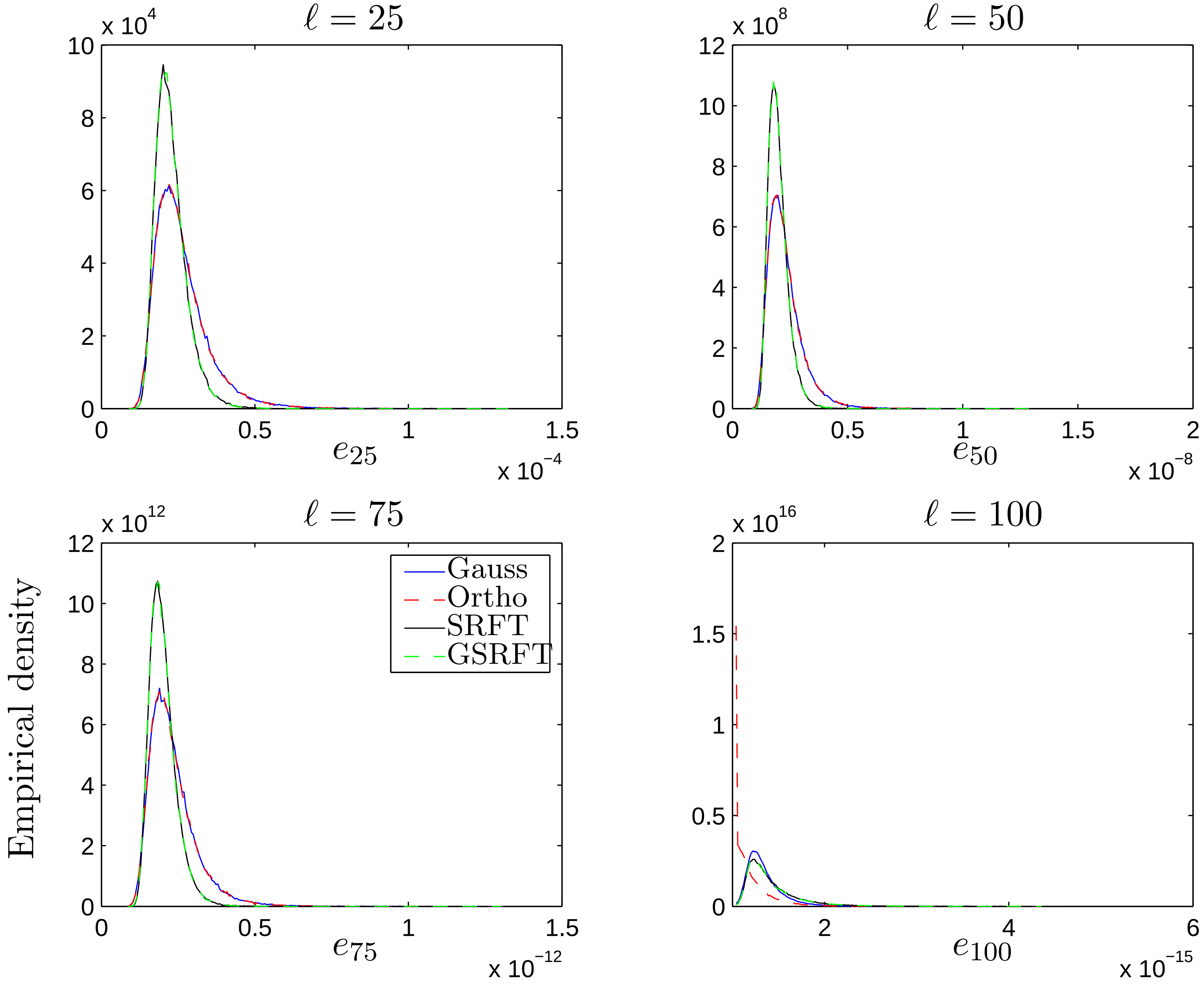


Note: The development of an error estimator resolves the issue of not knowing the numerical rank in advance!

Was this just a lucky realization?

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

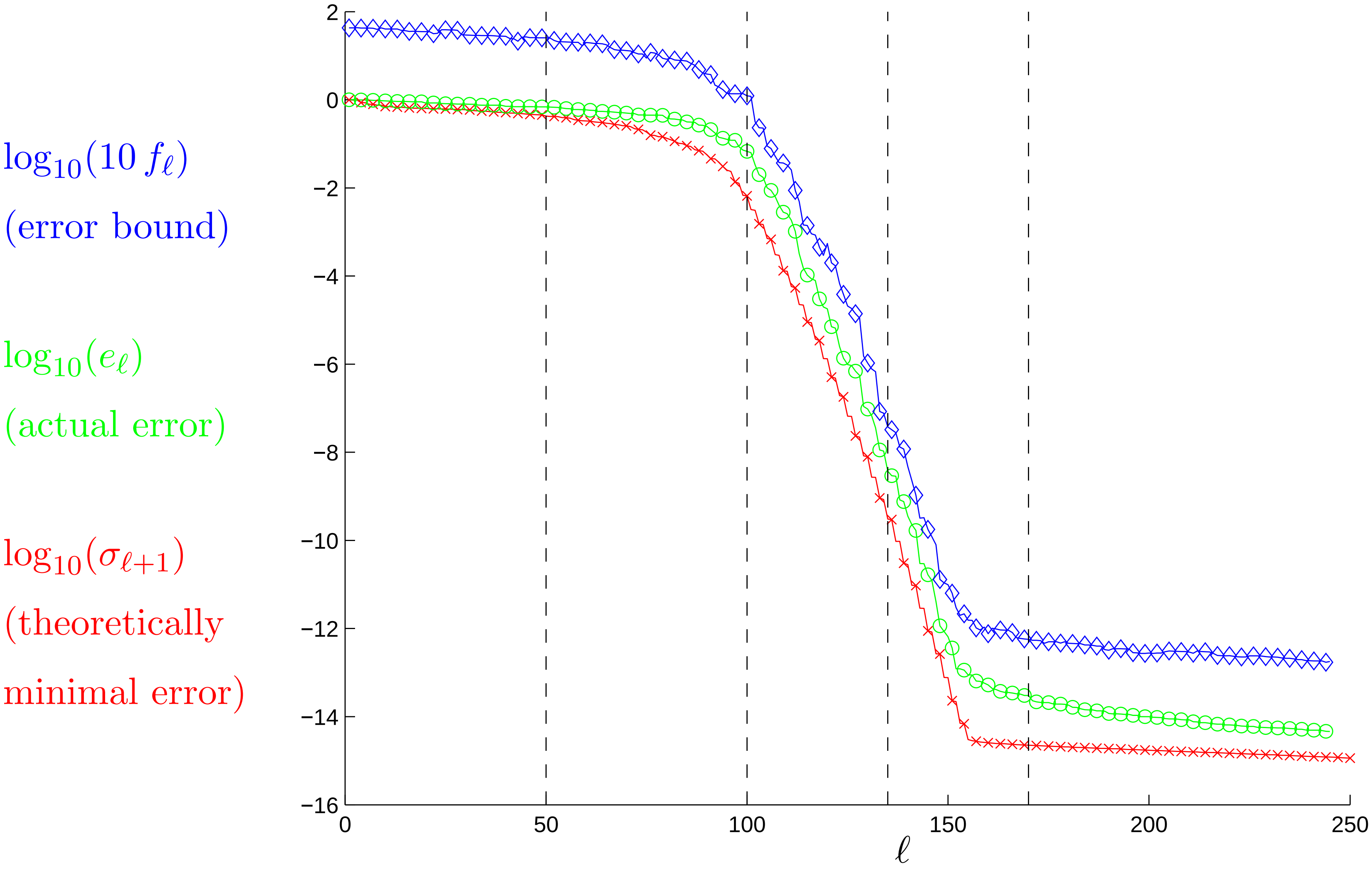


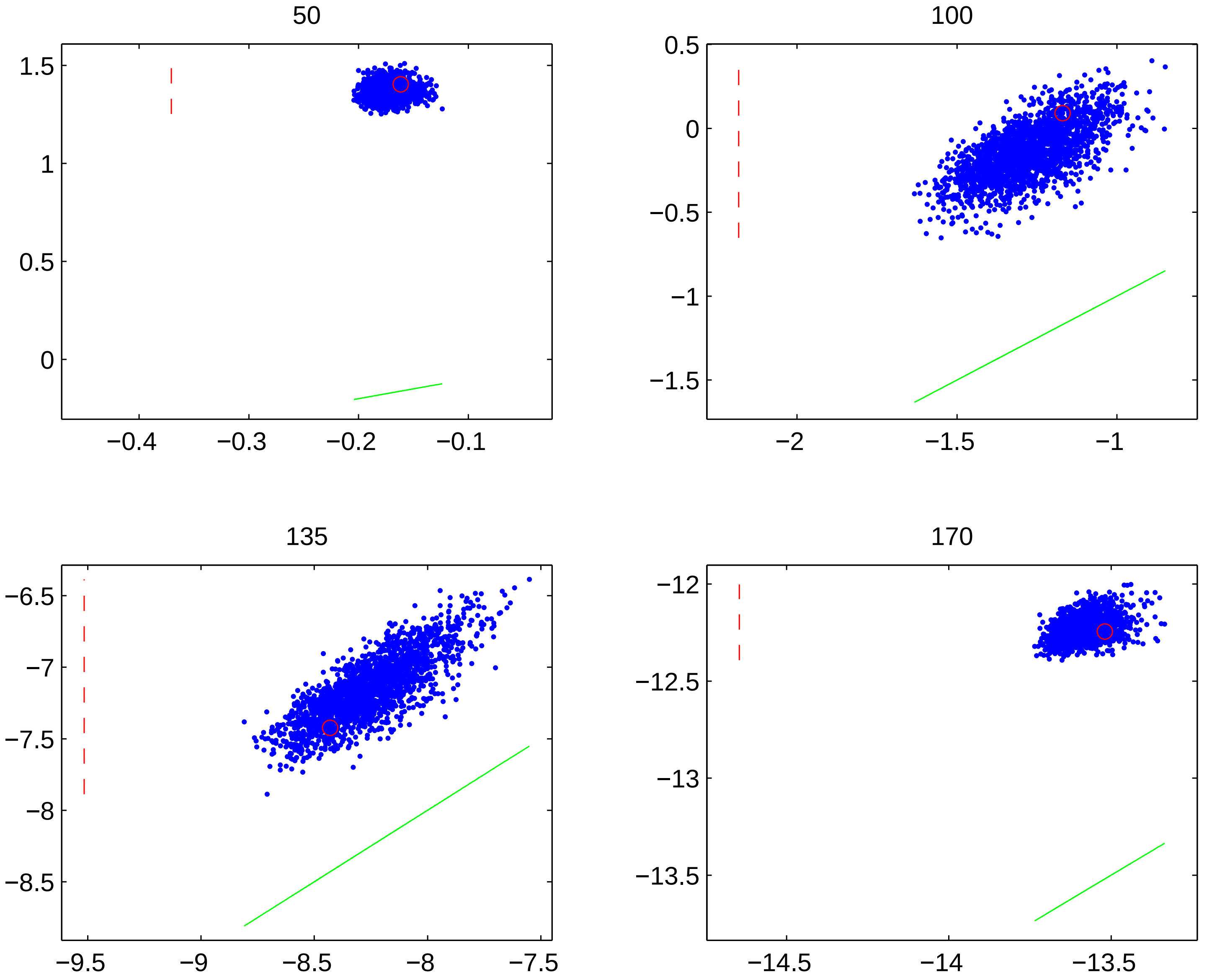

$\ell = 25$
x 10^4
10
8
6
4
2
0
0
0.5
1
1.5
e_{25}
x 10^-4
$\ell = 50$
x 10^8
12
10
8
6
4
2
0
0
0.5
1
1.5
2
e_{50}
x 10^-8
$\ell = 75$
x 10^12
12
10
8
6
4
2
0
0
0.5
1
1.5
e_{75}
x 10^-12
Gauss
Ortho
SRFT
GSRFT
Empirical density
$\ell = 100$
x 10^16
2
1.5
1
0.5
0
2
4
6
e_{100}
x 10^-15

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)




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 $\boldsymbol{x} \mapsto \mathsf{A}\,\boldsymbol{x}$ can be evaluated rapidly

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

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

2. Form the $m \times \ell$ sample matrix $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.
   Cost is $O(\ell\,(m+n))$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $\mathsf{Y}_\ell = \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{Y}_\ell$.
   Cost is $O(\ell^2\,m)$.

The asymptotic cost is the same as that of Arnoldi/Lanczos 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 $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.
   Cost is $O(\ell\, m\, n)$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $\mathsf{Y}_\ell = \mathsf{Q}_\ell\, \mathsf{Q}_\ell^{*}\, \mathsf{Y}_\ell$.
   Cost is $O(\ell^2\, m)$.

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

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

Let us change the random matrix Ω (as proposed by [Liberty, Rokhlin, Tygert, Woolfe 2006]):

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

2. Form the $m \times \ell$ sample matrix $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.
   Cost is $O(m\,n\,\log(\ell))$.

3. Construct an $m \times \ell$ orthonormal matrix Q_ℓ such that $\mathsf{Y}_\ell = \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{Y}_\ell$.
   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 $O(m\,n\,\log(k))$!

(Recall that the "post-processing" to obtain, say, an SVD costs $O((m+n)k^2)$.)

What is an “SRFT”?

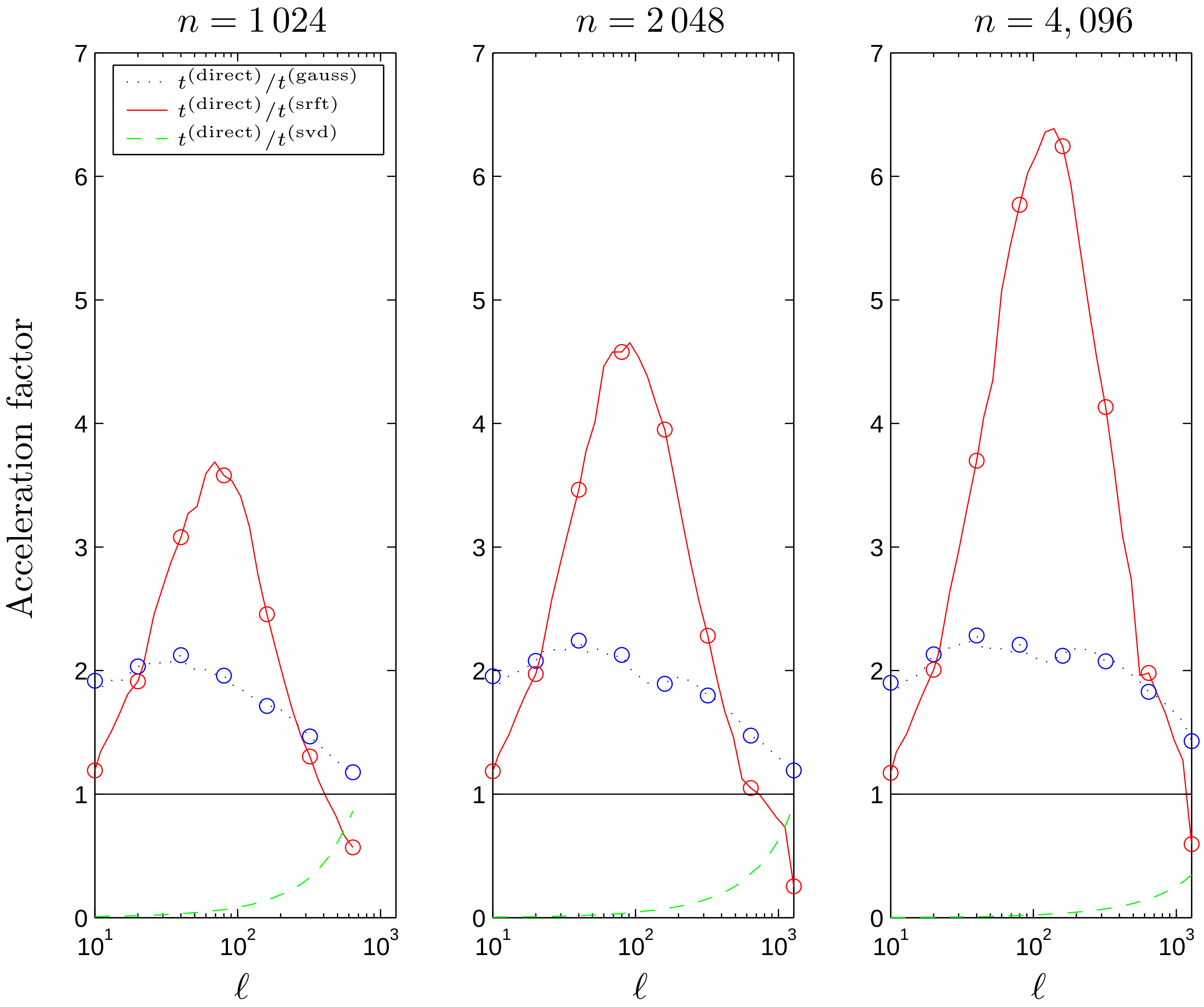
A Subsampled Random Fourier Transform. A random matrix with structure.

One possible choice:

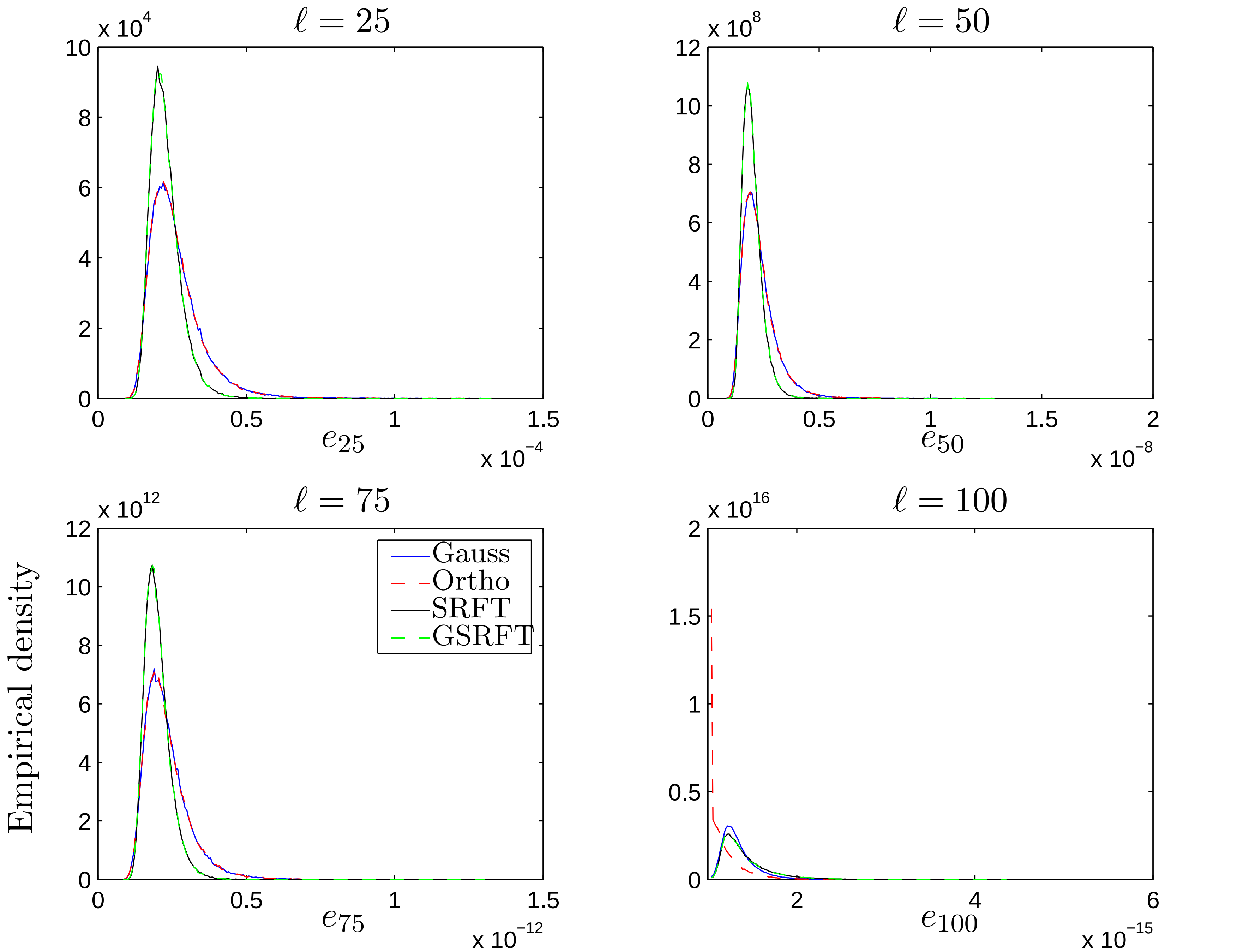
$$\begin{array}{ccccc} \Omega_\ell & = & \mathsf{D} & \mathsf{F} & \mathsf{S} \\ n\times\ell & & n\times n & n\times n & n\times\ell \end{array}$$

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, $\mathsf{F}_{jk} = \dfrac{1}{\sqrt{n}}\, e^{-2\pi i(j-1)(k-1)/n}$.
- 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 $\mathsf{D}\,\mathsf{F}$.)



$n = 1\,024$
$n = 2\,048$
$n = 4,096$
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
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

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.)

- Current theoretical results seem to overstate the risk of inaccurate results, at least in typical environments.

- 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 $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.
   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 $\mathsf{Y}_\ell = \mathsf{Q}_\ell\, \mathsf{Q}_\ell^*\, \mathsf{Y}_\ell$.
   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 $\mathsf{Y}_\ell = \mathsf{A}\,\Omega_\ell$.
   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 $\mathsf{Y}_\ell = \mathsf{Q}_\ell\,\mathsf{Q}_\ell^*\,\mathsf{Y}_\ell$.
   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.

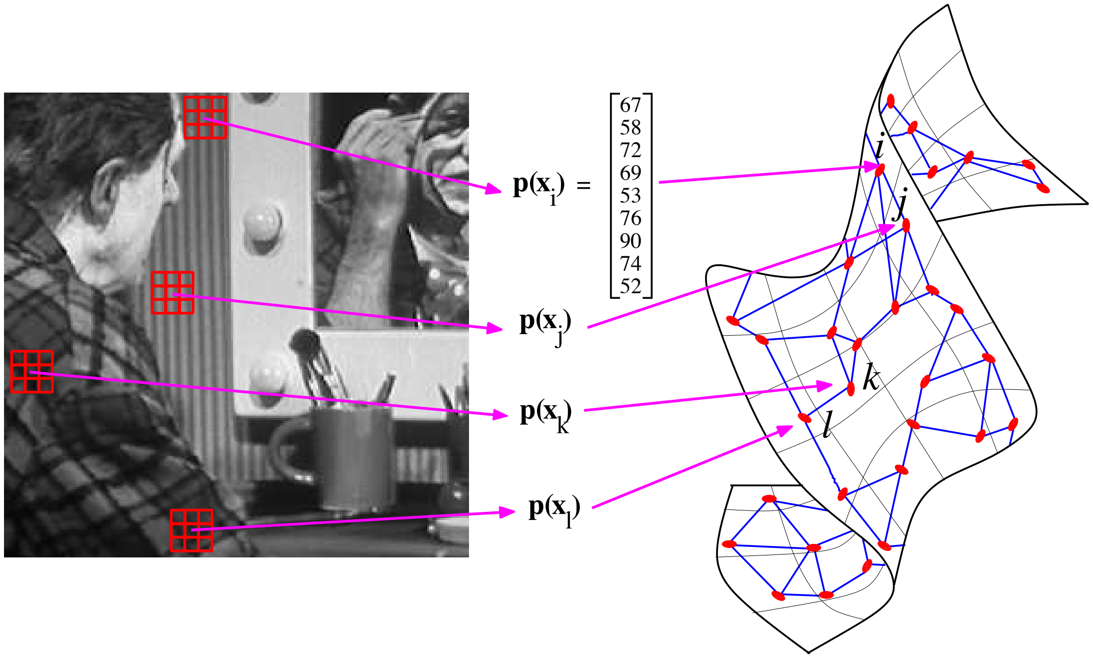
Similar considerations apply to implementation on multi-core architectures, streaming data, *etc*.

Problem: The basic algorithm can be very inaccurate!

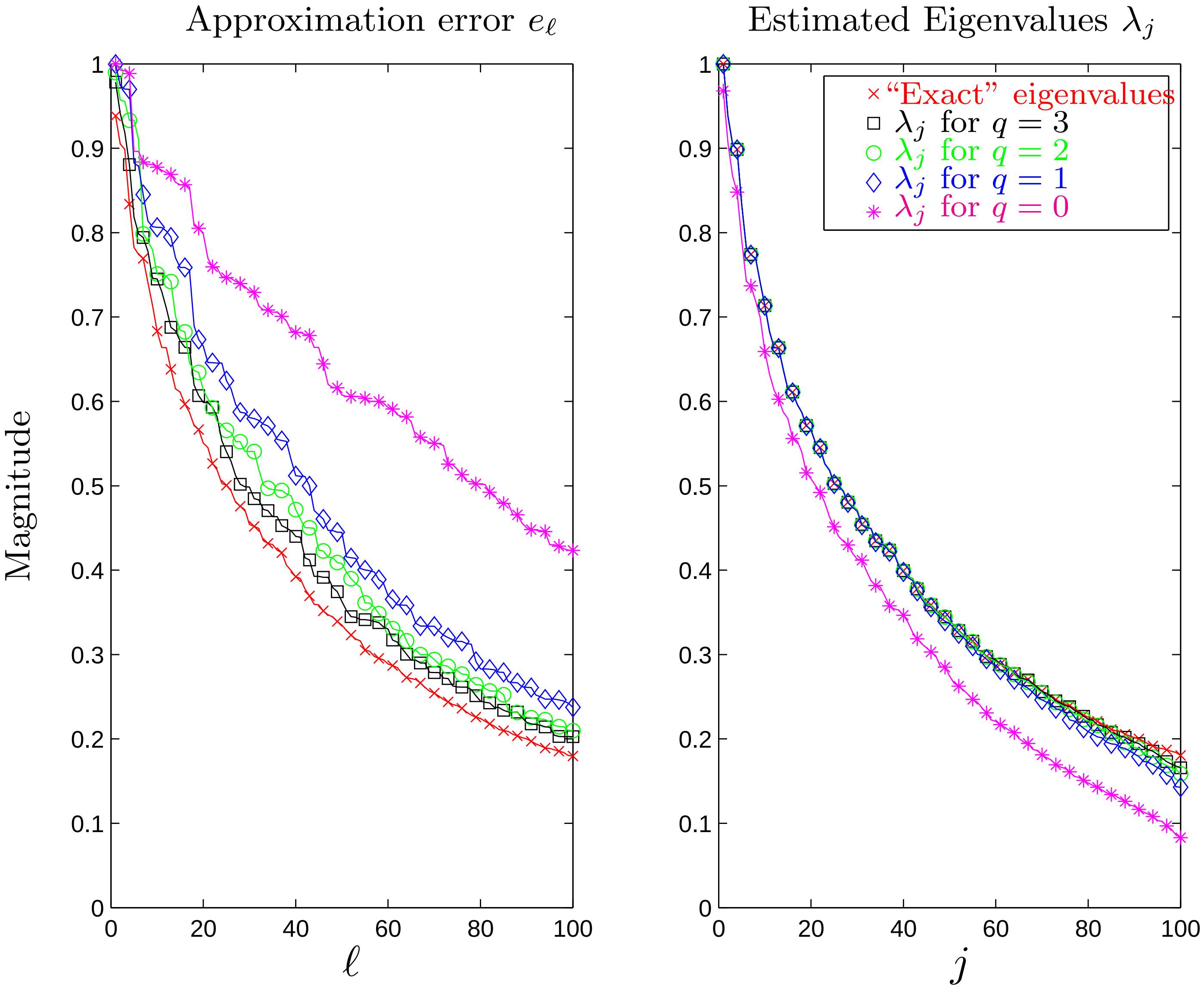
Example 1:

The matrix A being analyzed is a 9025 × 9025 matrix arising in a diffusion geometry approach to image processing.

To be precise, A is a graph Laplacian on the manifold of 3 × 3 patches.


$p(x_i) = \begin{bmatrix}67\\58\\72\\69\\53\\76\\90\\74\\52\end{bmatrix}$
$p(x_j)$
$p(x_k)$
$p(x_l)$
i
j
k
l

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



The pink lines reflect the output of the "basic" randomized sampling scheme. The errors (shown in the left graph) are intolerably large.

The problem is that singular vectors corresponding to modes associated with singular values under the threshold pollute the directions of our basis vectors.

We can quantify quite sharply how much. For instance:

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 $\mathsf{Y} = \mathsf{A}\Omega$, and let Q denote an orthonormal matrix such that Ran$(\mathsf{Q}) =$ Ran(Y). Then*

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

Moreover,

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

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 $\mathsf{Y} = \mathsf{A}\,\Omega$, and let Q denote an orthonormal matrix such that Ran(Q) = Ran(Y). For all $u, t \geq 1$,*

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

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,

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

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

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 $\mathsf{B} = (\mathsf{A}\,\mathsf{A}^*)^q\,\mathsf{A}$ has the same left singular vectors as A, and its singular values are

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

Much faster decay — so use the sample matrix

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

instead of

$$\mathsf{Y} = \mathsf{A}\,\Omega.$$

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 $\mathsf{B} = (\mathsf{A}^*\mathsf{A})^q\mathsf{A}$, and construct the sample matrix $\mathsf{Z} = \mathsf{B}\,\Omega$. Let P_Z denote the orthogonal projector onto the range of Z. Then*

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

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

$$||(\mathsf{I} - \mathsf{P}_\mathsf{Y})\,\mathsf{A}|| \leq C\,\sigma_{k+1},$$

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

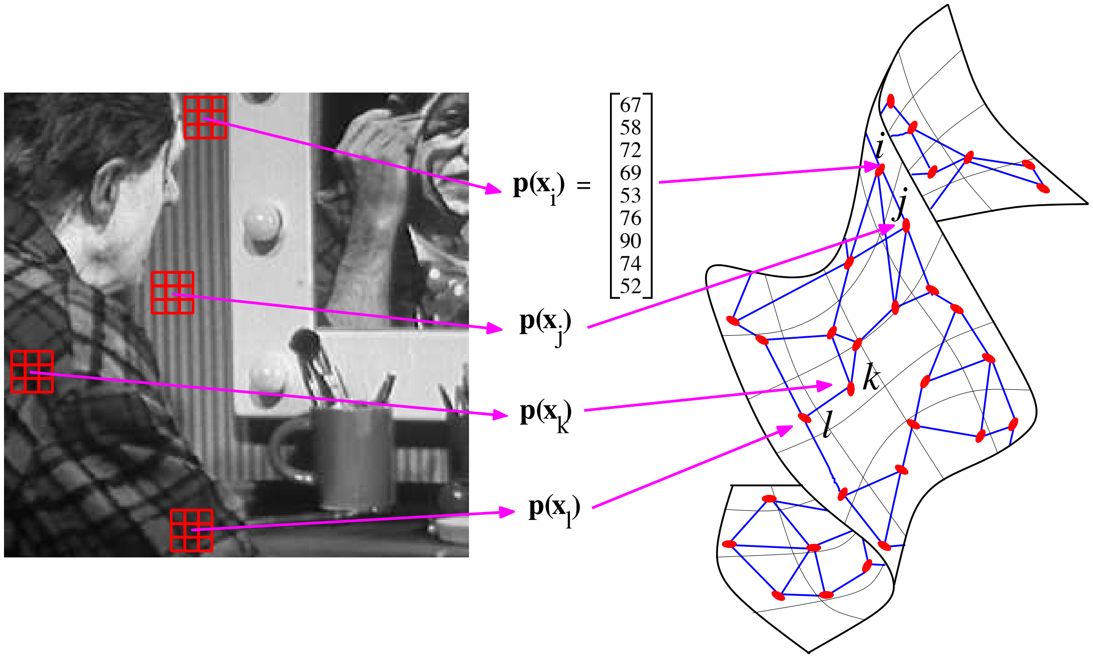
$$||(\mathsf{I} - \mathsf{P}_\mathsf{Z})\,\mathsf{A}|| \leq C^{1/(2\,q+1)}\,\sigma_{k+1}$$

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

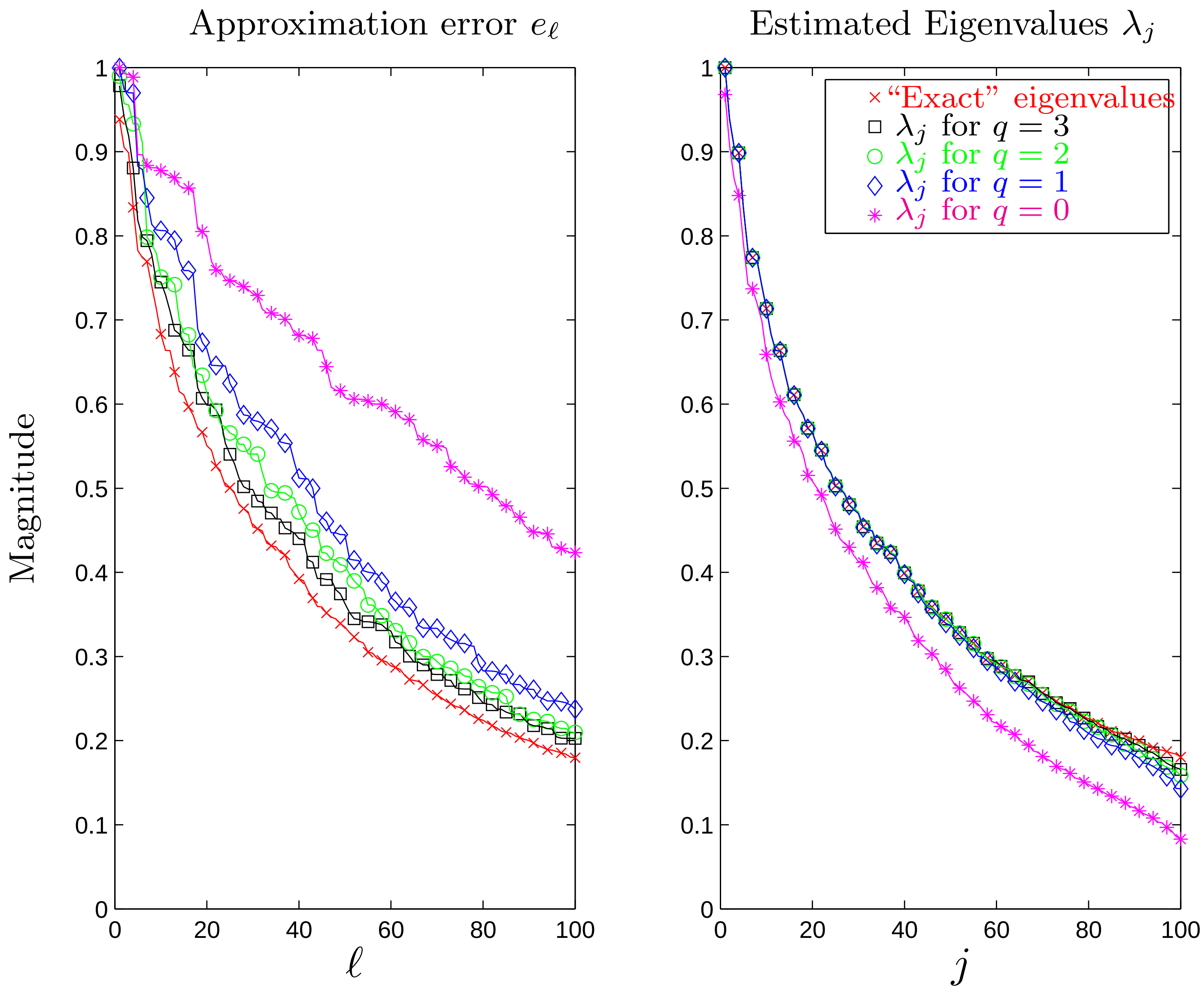
Example 1:

The matrix A being analyzed is a 9025 × 9025 matrix arising in a diffusion geometry approach to image processing.

To be precise, A is a graph Laplacian on the manifold of 3 × 3 patches.


$\mathbf{p}(\mathbf{x}_i) = \begin{bmatrix} 67 \\ 58 \\ 72 \\ 69 \\ 53 \\ 76 \\ 90 \\ 74 \\ 52 \end{bmatrix}$
$\mathbf{p}(\mathbf{x}_j)$
$\mathbf{p}(\mathbf{x}_k)$
$\mathbf{p}(\mathbf{x}_l)$
i
j
k
l

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



Observe how rapidly the error improves as q is increased!

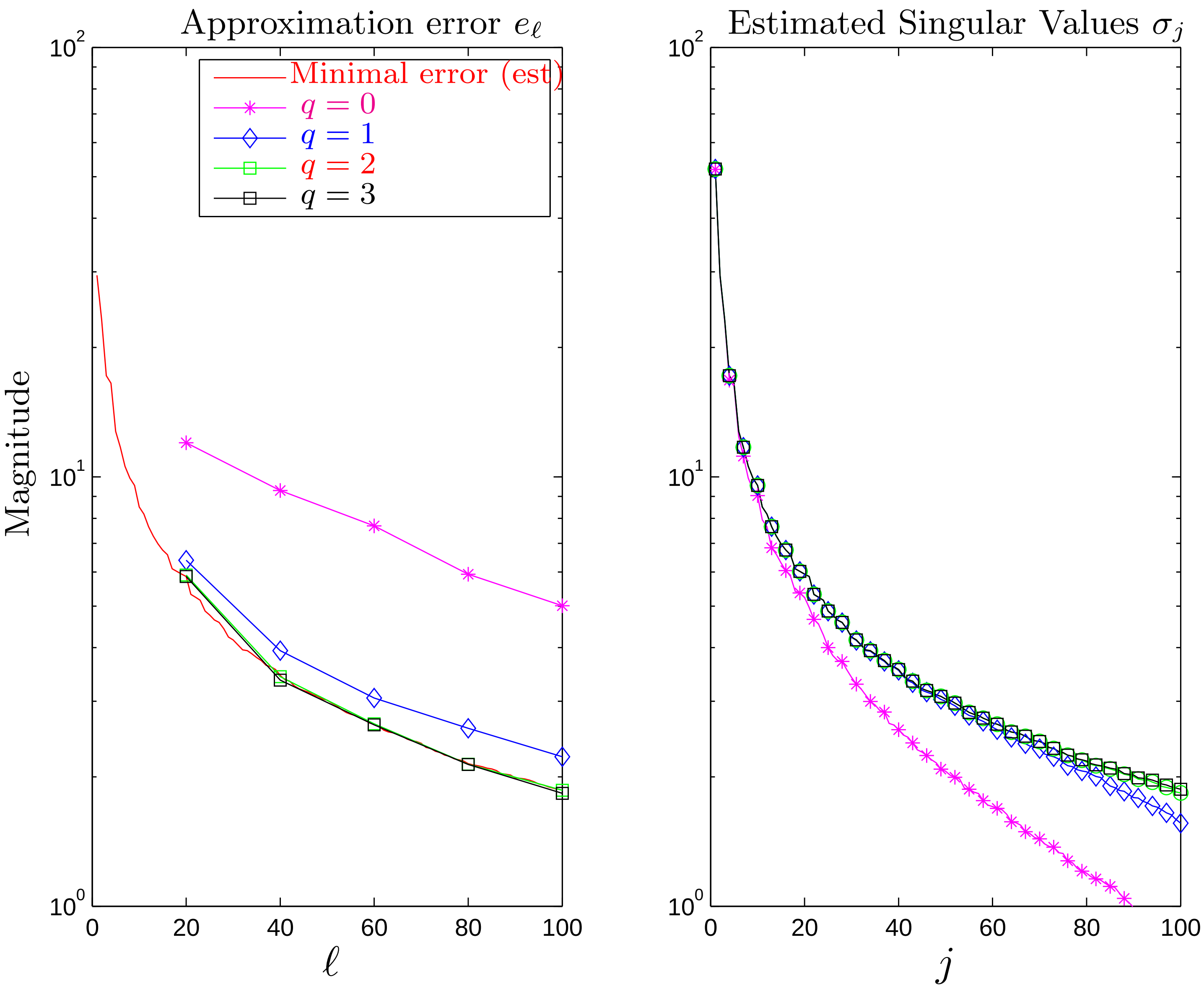
Example 2: “Eigenfaces”

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 A.

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



Again, observe how rapidly the error improves as q is increased!

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, ...)

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.

- In important environments, randomized methods outperform classical methods in terms of speed, accuracy, and robustness.

- The risk that the methods "fail" can inexpensively be rendered negligible (10^{-10}, 10^{-20}, 10^{-50}, $\dots$).

- Perhaps the most important benefit of randomized methods is that they vastly reduce communication constraints:
  - Low-pass methods. Sometimes single pass! Streaming data.
  - Out of core execution. Data stored on disk.
  - Parallel implementation.

- Excellent error control. Close to theoretically minimal errors are achievable.

- For large scale SVD/PCA, these algorithms are highly recommended; they compare favorably to existing methods in almost every regard. Free software can be downloaded → google *Mark Tygert*

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