Fast Matrix Computations via Randomized Sampling

Gunnar Martinsson, The University of Colorado at Boulder

Ph.D. Students:	Adrianna Gillman
	Nathan Halko
	Patrick Young
Collaborators:	Edo Liberty
	Vladimir Rokhlin
	Joel Tropp
	Mark Tygert
	Franco Woolfe

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:

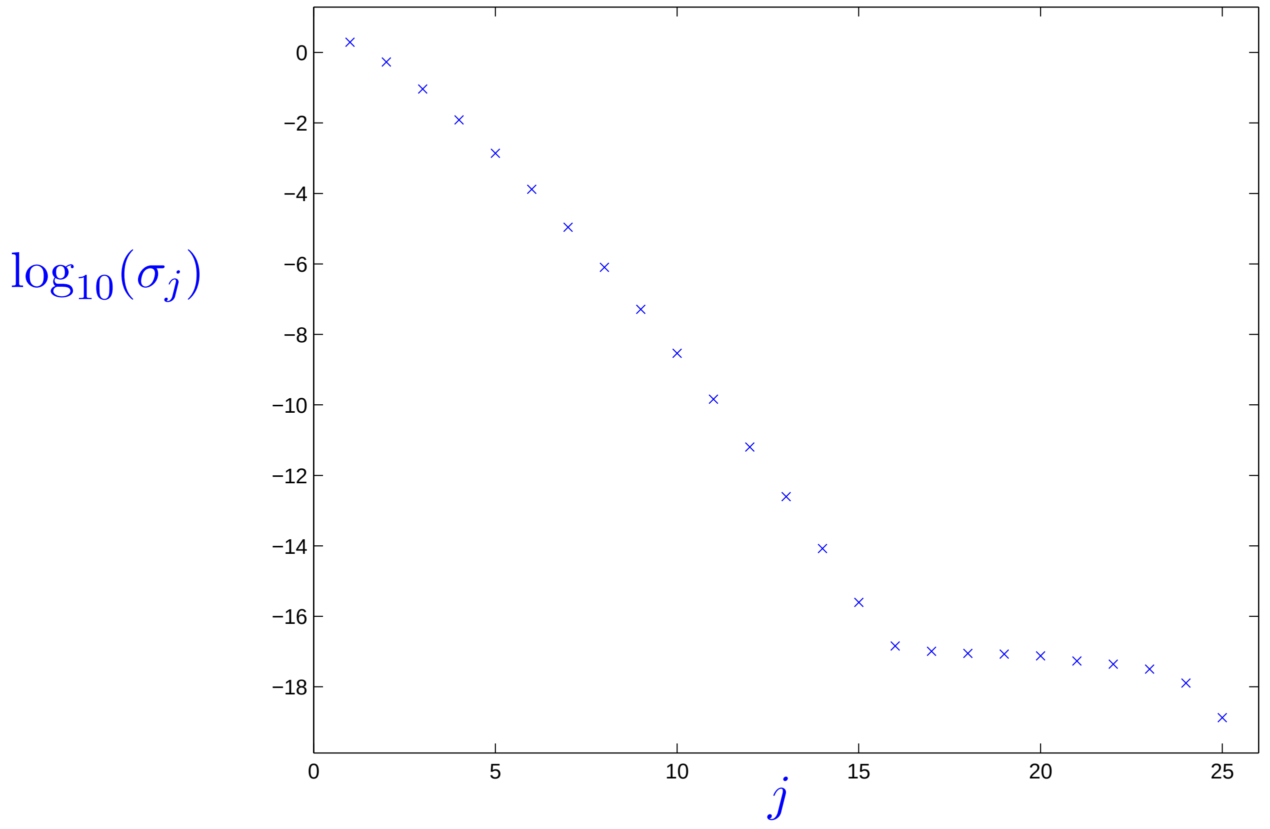
$$\sigma_j = \inf_{\mathrm{rank}(\tilde{A})=j-1} ||A - \tilde{A}||.$$

and

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

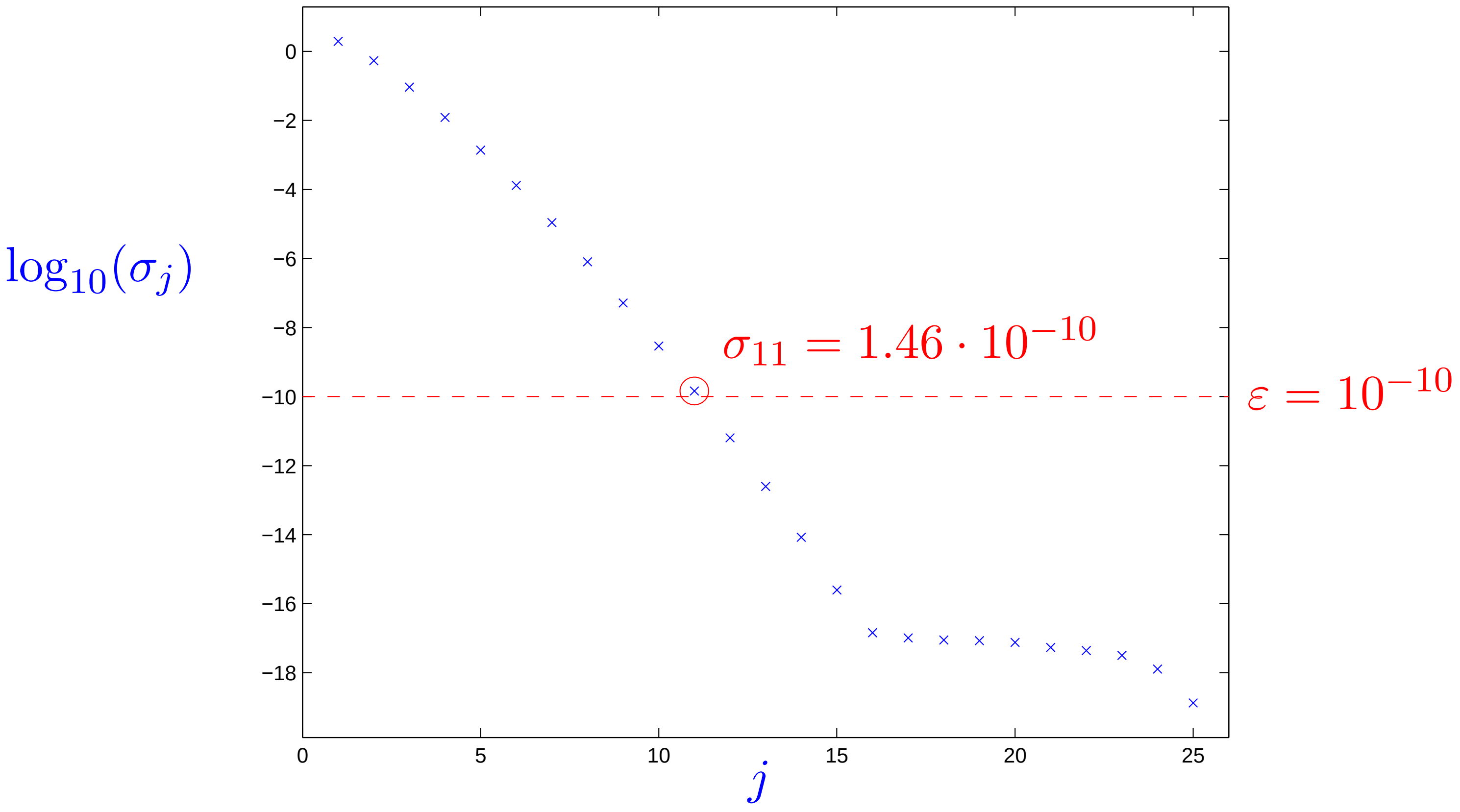
The decay of the singular values determines how well a matrix can be approximated by low-rank factorizations.

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.

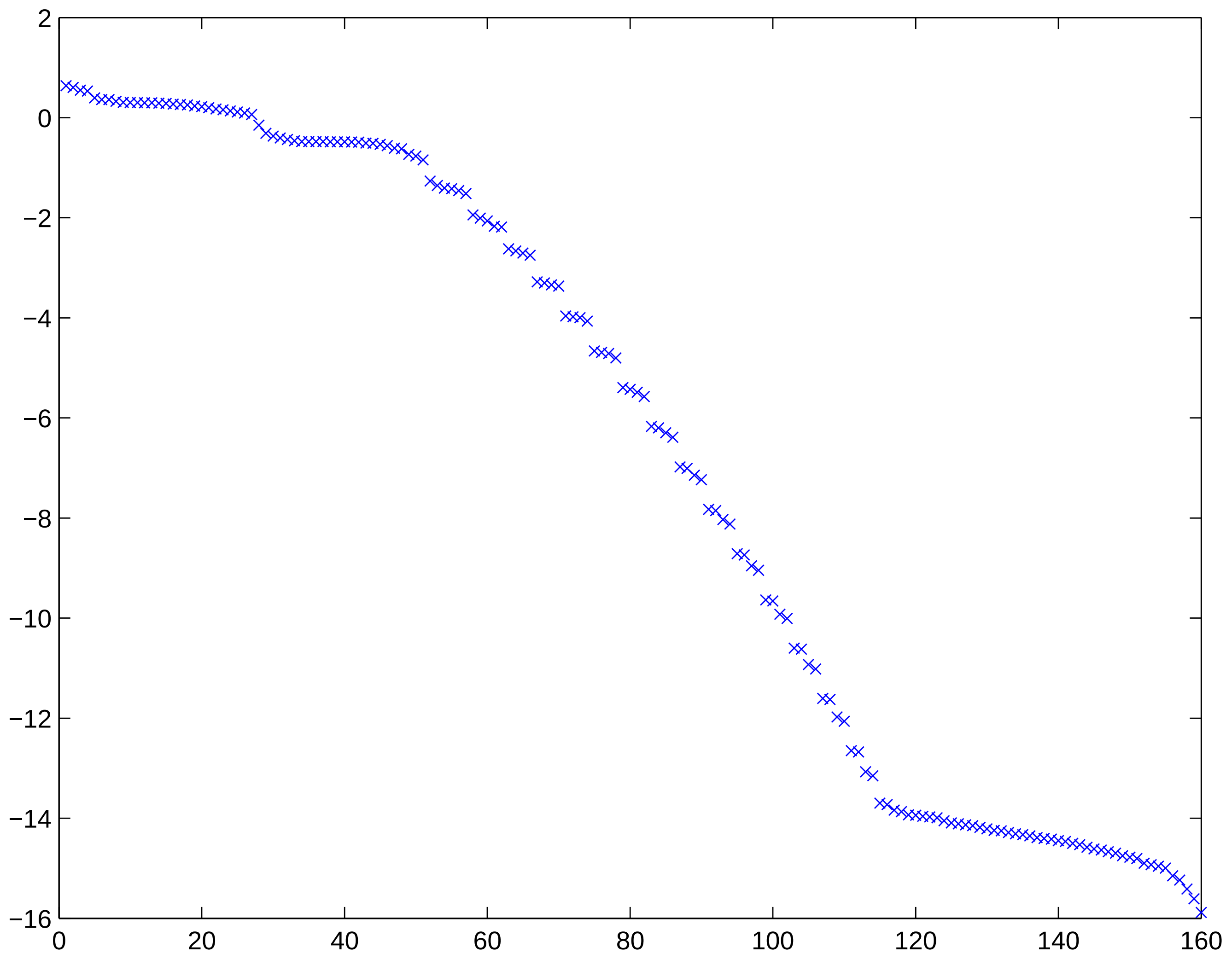
Example: Let A be a 25×25 Hilbert matrix, *i.e.* $A_{ij} = 1/(i+j-1)$. Let σ_j denote the j'th singular value of A.



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

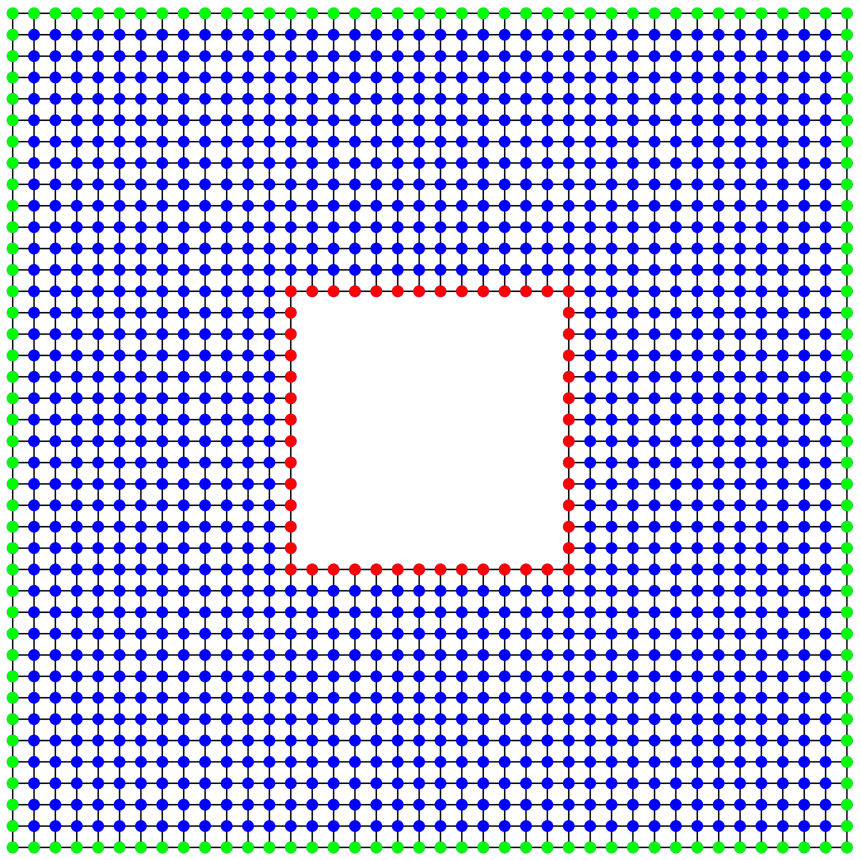
Question: Are there interesting examples of matrices whose singular values decay?

Many of the operators of classical mathematical physics are compact with exponentially decaying spectra.



“Scattering matrix” for 2D acoustic scattering between two squares.

This is one reason why multiscale analysis (a.k.a. "model reduction", "coarse graining", *etc*) works. Consider a potential problem on this electric network:

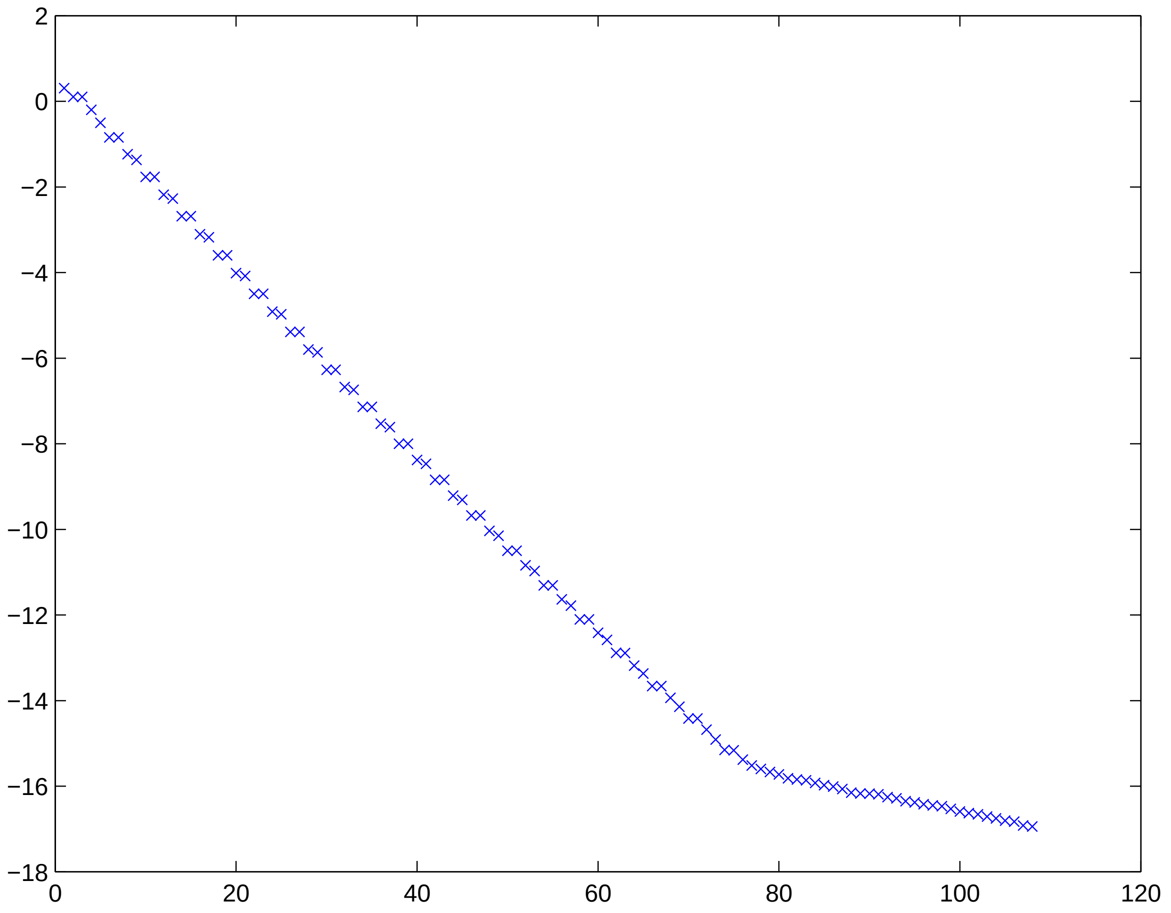


The green nodes are grounded. (Potential is kept at zero.)

At the red nodes, an electric flux f is applied.

Let A denote the linear map such that $A\,f$ is the resulting flux at the green nodes.

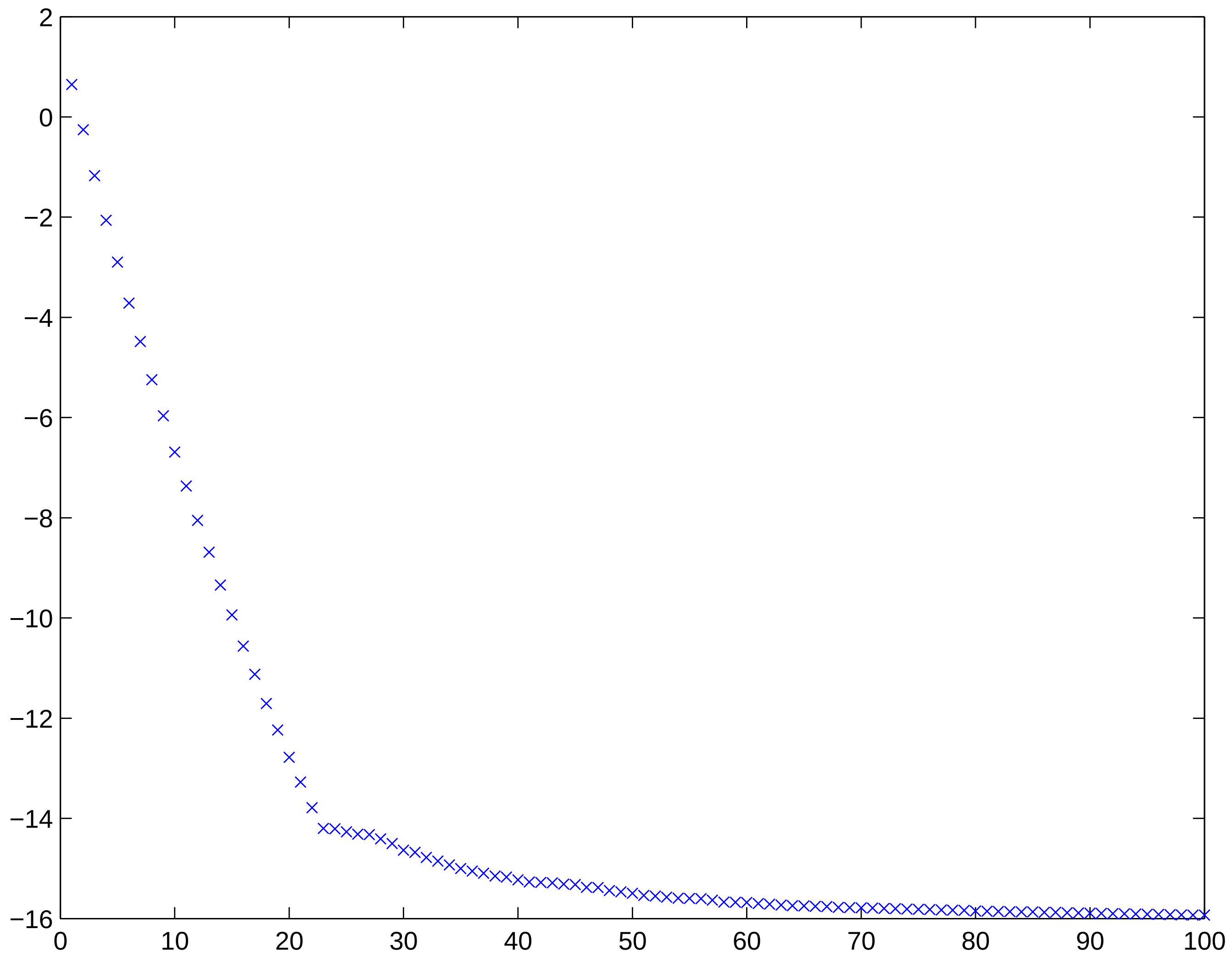
We recall from the previous slide:
The matrix A maps fluxes on the inner boundary to fluxes on outer boundary.



Singular values of A for a lattice twice the size of the one on the previous slide.

(What we see is a manifestation of a Saint Venant principle for the lattice.)

Example from numerical analysis:



Spectrum of a submatrix of the inverse of a large sparse matrix resulting upon discretization of Laplace's equation using the Finite Element Method:

Example: Population genetics

From: P. Paschou, E. Ziv, E. Burchard, M.W. Mahoney, and P. Drineas

Matrix A recording "Single Nucleotide Polymorphisms" (SNPs).
We study a sequence of base pairs that take on the values, say, "R" and "S".
Matrix entry A_{ij} records the value for person j of an allele pair i:

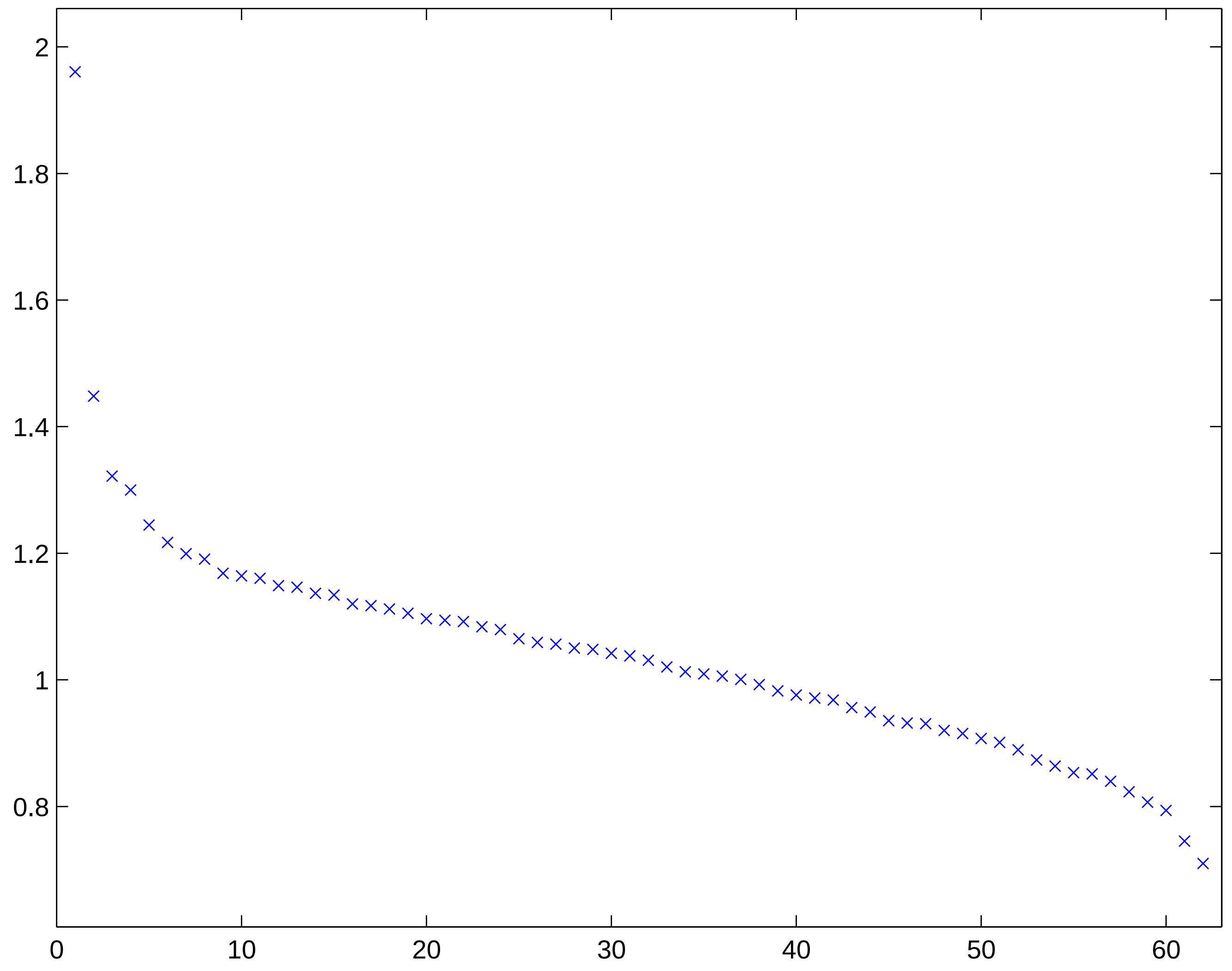
$$\begin{array}{lcll} \text{genotype SS} & \sim & A_{ij} = & -1 \\ \text{genotype RS} & \sim & A_{ij} = & 0 \\ \text{genotype RR} & \sim & A_{ij} = & 1 \end{array}$$

In other words

	Person 1	Person 2	Person 3	$\cdots$
Allele 1	RS	RR	RS	$\cdots$
Allele 2	SS	SS	SS	$\cdots$
Allele 3	RS	RS	SS	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

$$\sim \qquad A = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ -1 & -1 & -1 & \cdots \\ 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

Singular values of a 400×62 matrix A (400 alleles, 62 people):



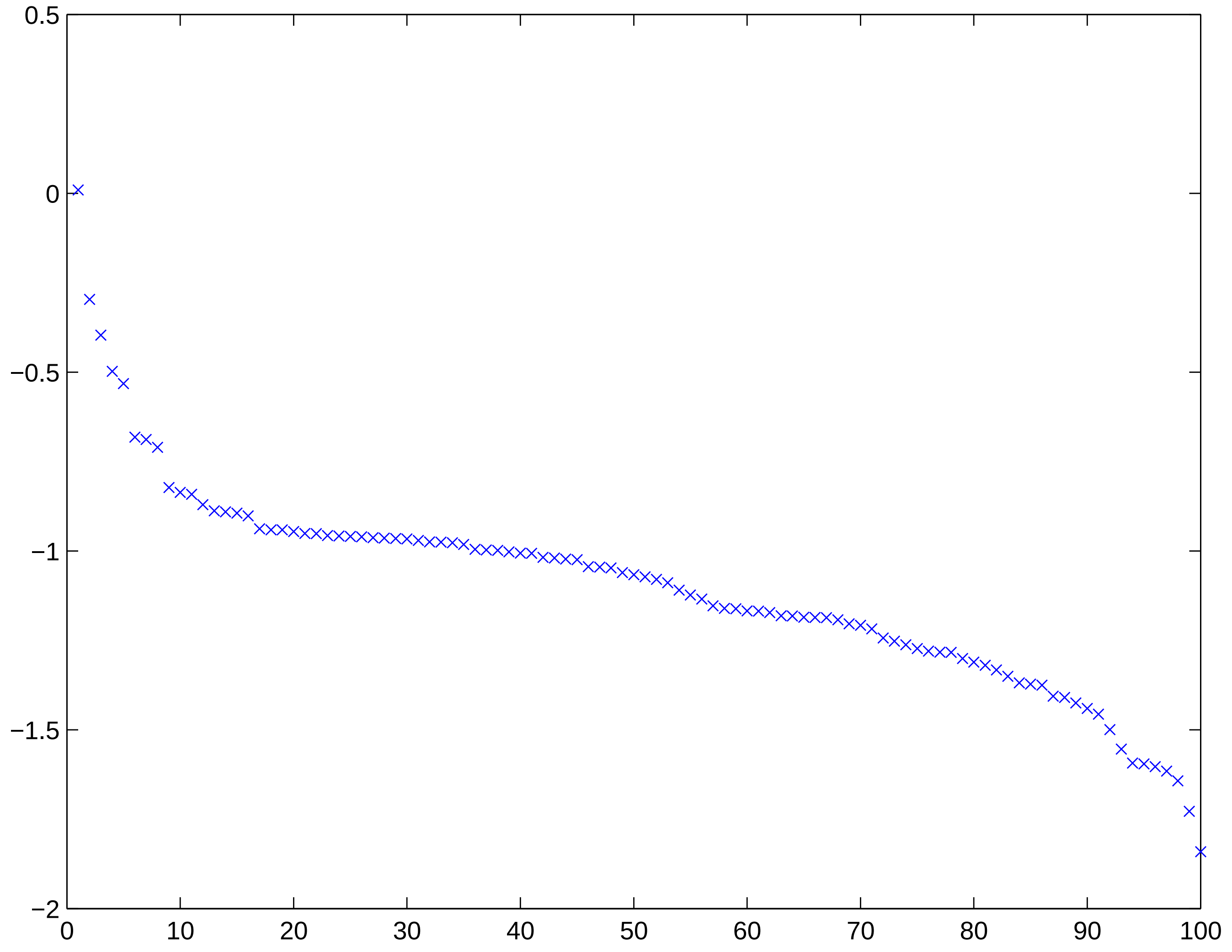
Clustering information can be extracted from the first few singular vectors.

This type of spectrum is typical for matrices arising from statistical samples.

- Genomics.
- Financial data.
- Image analysis.
- Network analysis — clustering, search
  (*cf.* “page rank”-type algorithms used by Google).
- . . .

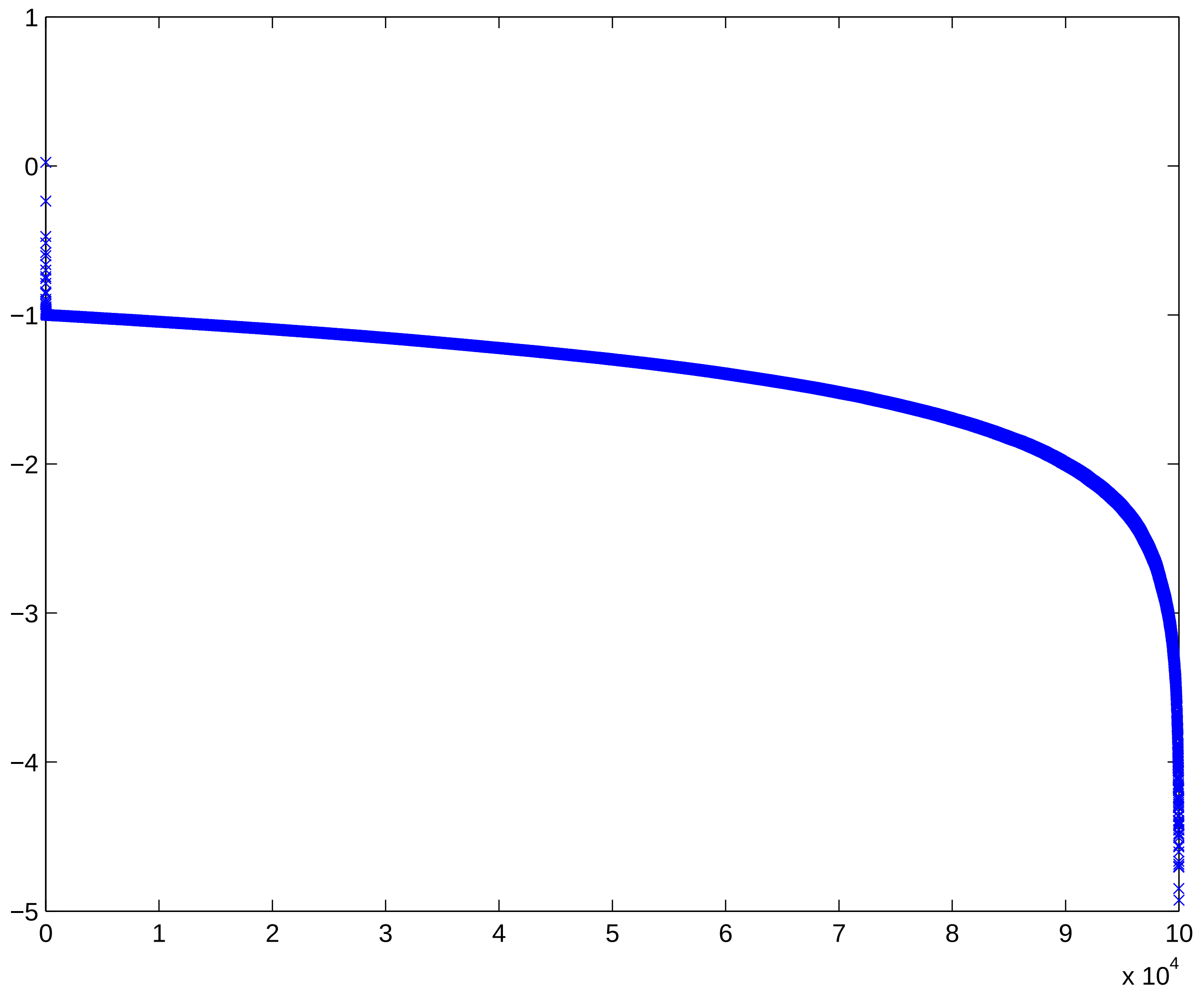
In this context, the term “Principal Component Analysis” (PCA) is frequently used.

There is a snake in the garden, however.



While it is easy to compute PCAs of small matrices (Matlab works great) . . .

…typical matrices in this context are *large*.



In this case, almost all the "mass" of the matrix is noise.

Picking up the "signal" is then a very hard problem in numerical analysis.

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 $||A - Q\,Q^{\rm t}\,A|| \leq \varepsilon$.

Notes:

- Once Q has been constructed, it is in many environments possible to construct standard factorization (such as the SVD) 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 $||A - Q\,Q^{\rm t}\,A|| \leq \varepsilon$.

We will discuss two environments:

Case 1:

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

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

The new methods are also $O(T_{\rm mult}\,k)$ 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 specified 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 a vector $\omega \in \mathbb{R}^N$ is a Gaussian random vector if

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix},$$

where the numbers $\{\omega_j\}_{j=1}^N$ are random variables drawn independently from a normalized Gaussian distribution.

Note that the probability distribution of ω is isotropic in the sense that it is invariant under rotations in $\mathbb{R}^N$.

Note: In practise, the vectors ω 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.

Let $\omega_1,\,\omega_2,\,\dots$ be a sequence of Gaussian random vectors in $\mathbb{R}^N$.

Form the vectors

$$y_1 = A\,\omega_1, \qquad y_2 = A\,\omega_2, \qquad y_3 = A\,\omega_3, \qquad \dots$$

Each y_j is a "random linear combination" of columns of A.

If ℓ is an integer such that $\ell \geq k$, then there is a chance that the vectors

$$\{y_1,\,y_2,\,\dots,\,y_\ell\}$$

span the column space of A "to within precision ε". Clearly, the probability that this happens gets larger, the larger the gap between ℓ and k.

What is remarkable is how fast this probability approaches one.

How to measure “how well we are doing”:

Let $\omega_1, \omega_2, \ldots, \omega_\ell$ be the sequence of Gaussian random vectors in $\mathbb{R}^N$.

Set $y_j = A\,\omega_j$.

Orthonormalize the vectors $[y_1,\, y_2,\, \ldots\, ,\, y_\ell]$ and collect the result in the matrix Q_ℓ.

The “error” after ℓ steps is then

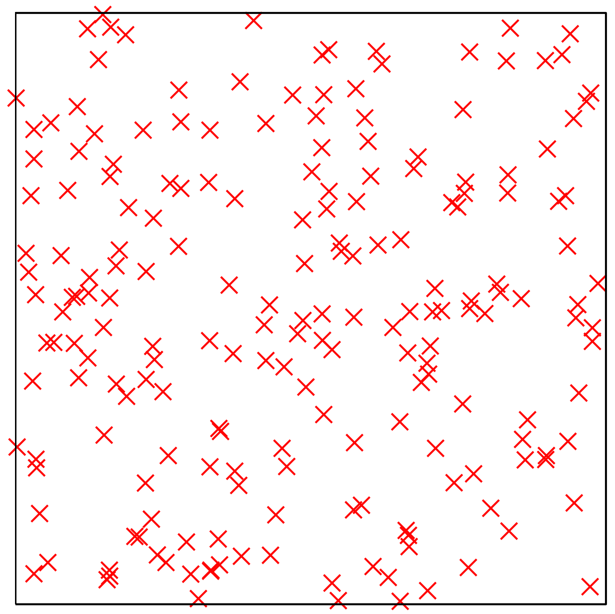
$$e_\ell = ||A - Q_\ell\, Q_\ell^{\mathrm{t}}\, A||$$

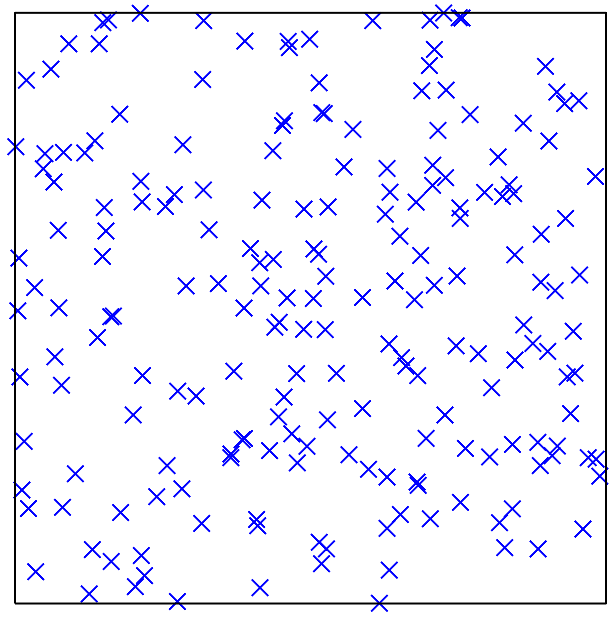
The quantity e_ℓ should be compared to the minimal error

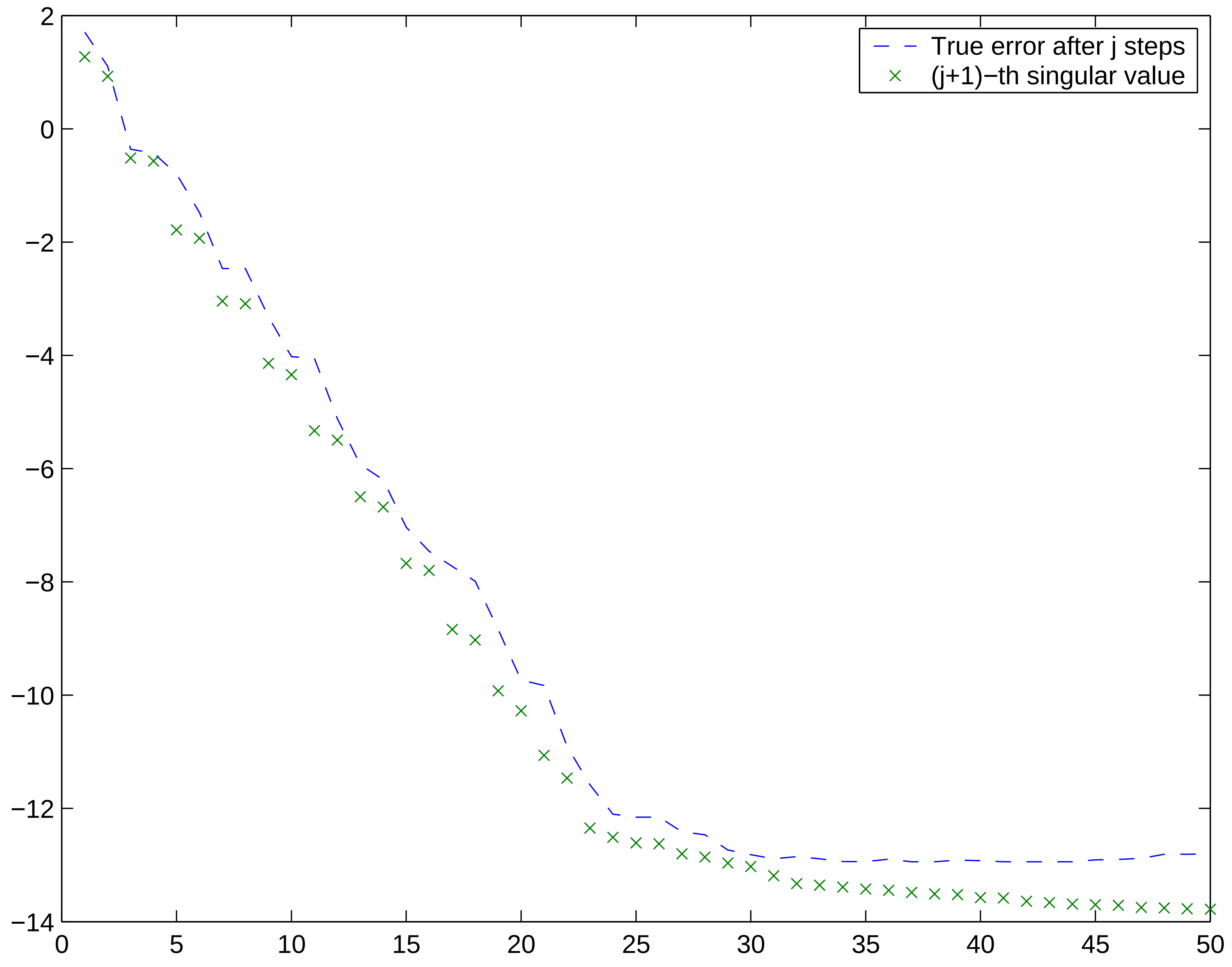
$$\sigma_{\ell+1} = \min_{\mathrm{rank}(B)=\ell} ||A - B||.$$

To illustrate the performance, we consider the following particular choice of A:

Let A be an $N \times N$ matrix with entries $A_{ij} = \log|z_i - w_j|$ where z_i and w_j are points in two separated clusters in $\mathbb{R}^2$.


sources z_i


targets w_j



$\varepsilon = 10^{-10}$, exact ε-rank $= 20$, nr. of matrix-vector multiplies required $= 21$.

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

Let $\omega_1, \omega_2, \dots, \omega_\ell$ be the sequence of Gaussian random vectors in $\mathbb{R}^N$.

Set $y_j = A\,\omega_j$.

Orthonormalize the vectors $[y_1,\, y_2,\, \dots\,,\, y_\ell]$ and collect the result in the matrix Q_ℓ.

The "error" after ℓ steps is then

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

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 $\omega_1,\, \omega_2,\, \dots,\, \omega_\ell$ be the sequence of Gaussian random vectors in $\mathbb{R}^N$.

Set $y_j = A\,\omega_j$.

Orthonormalize the vectors $[y_1,\, y_2,\, \dots\,,\, y_\ell]$ and collect the result in the matrix Q_ℓ.

The “error” after ℓ steps is then

$$e_\ell = ||A - Q_\ell\, Q_\ell^{\rm t}\, A|| = ||(I - Q_\ell\, Q_\ell^{\rm t})\, A||.$$

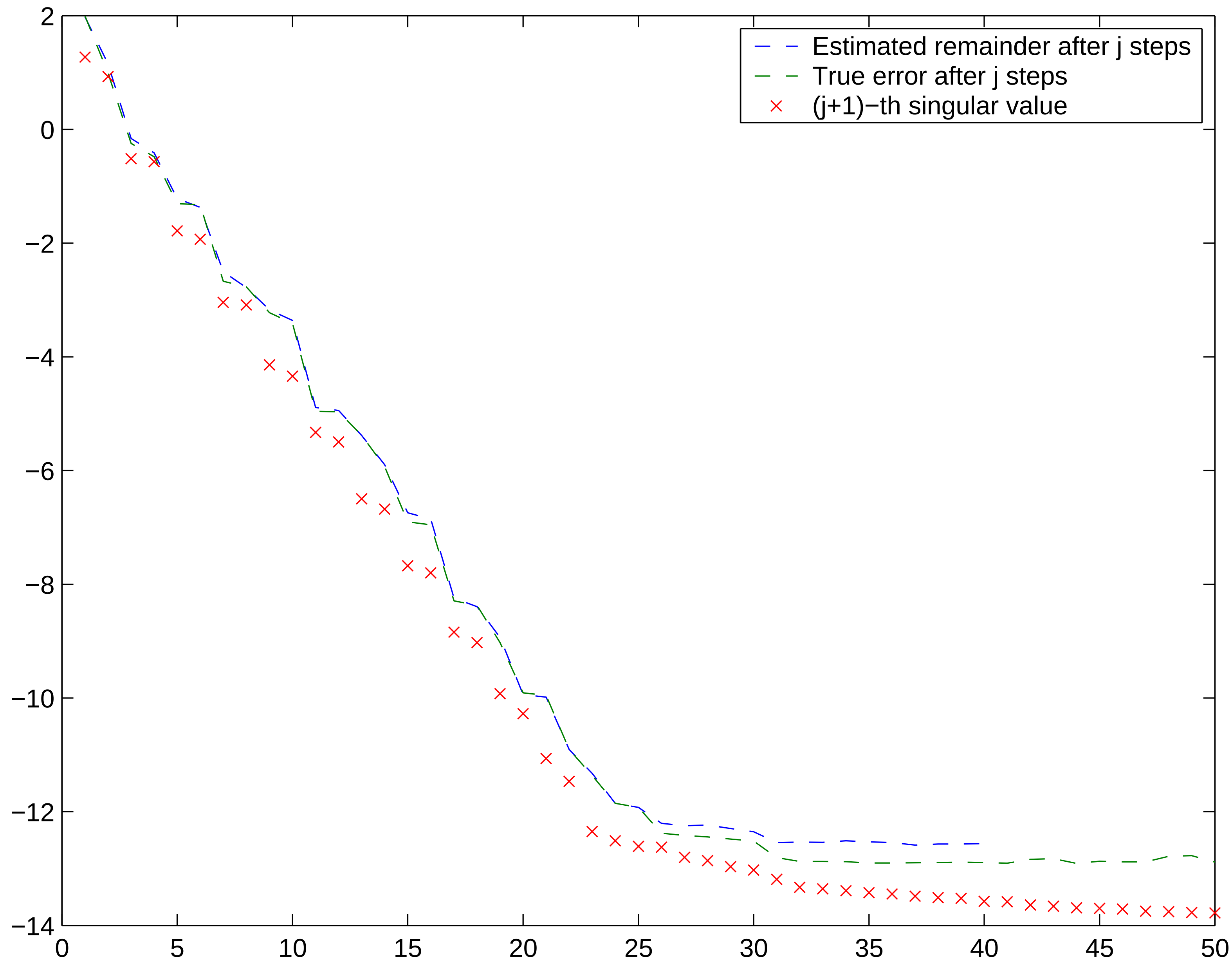
The quantity e_ℓ should be compared to the minimal error

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

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

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

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



$\varepsilon = 10^{-10}$, exact ε-rank $= 20$, nr. of matrix-vector multiplies required $= 21+10$.

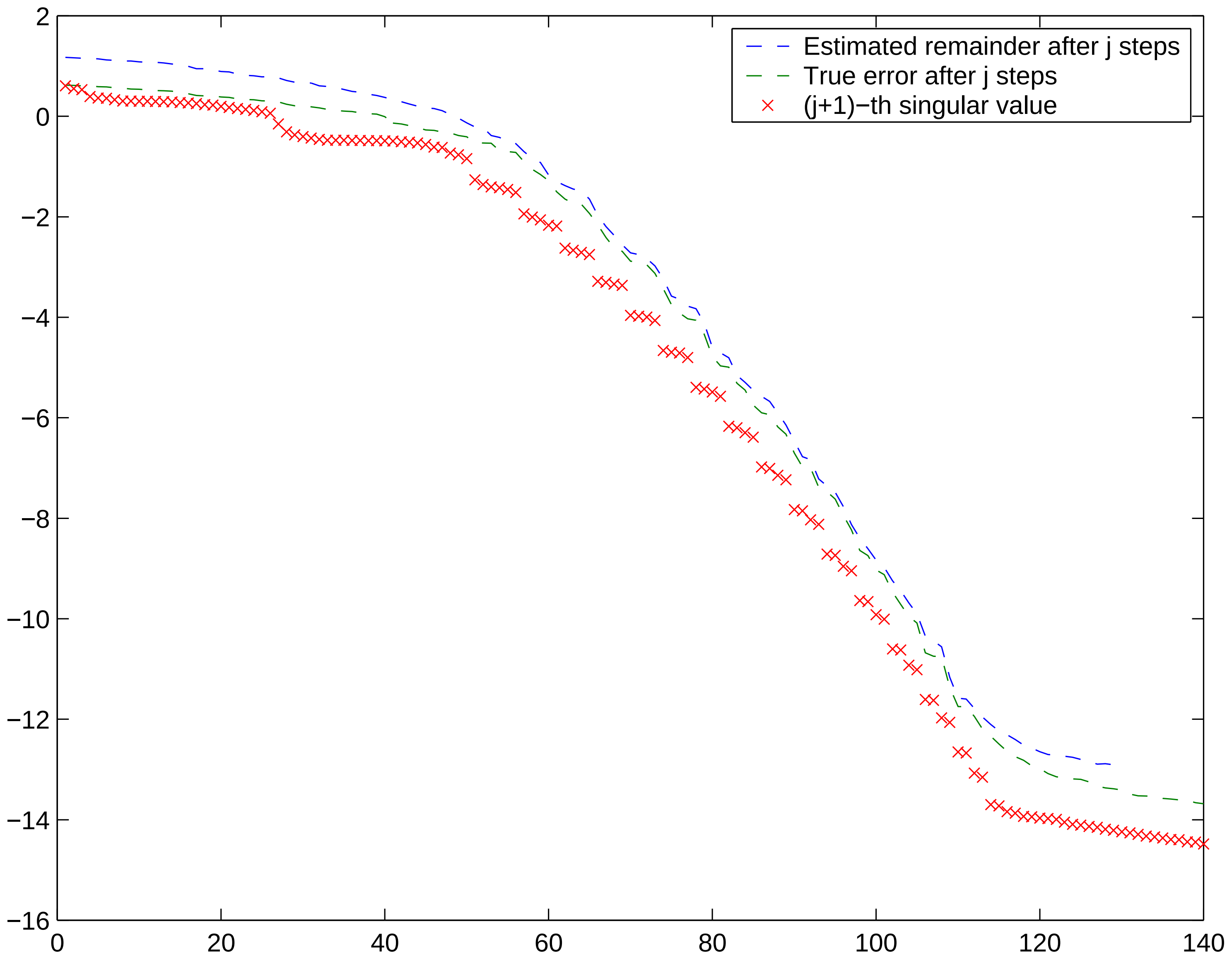
Was this just a lucky realization?

We collected statistics from 1 000 000 realizations:
(For a slightly different experiment — here the exact ε-rank is 34.)

Number of matrix-vector multiplies required:	Frequency:
34 (+10)	15063
35 (+10)	376163
36 (+10)	485124
37 (+10)	113928
38 (+10)	9420
39 (+10)	299
40 (+10)	3

Note: The post-processing correctly determined the rank to be 34 *every time*, and the error in the factorization was *always* less than 10^{-10}.

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



$\varepsilon = 10^{-10}$, exact ε-rank $= 101$, nr. of matrix-vector multiplies required $= 106$.

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

$$\begin{array}{cccc} Y & = & A & \Omega. \\ N \times \ell & & N \times N & N \times \ell \end{array}$$

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 $\mathbb{C}$.

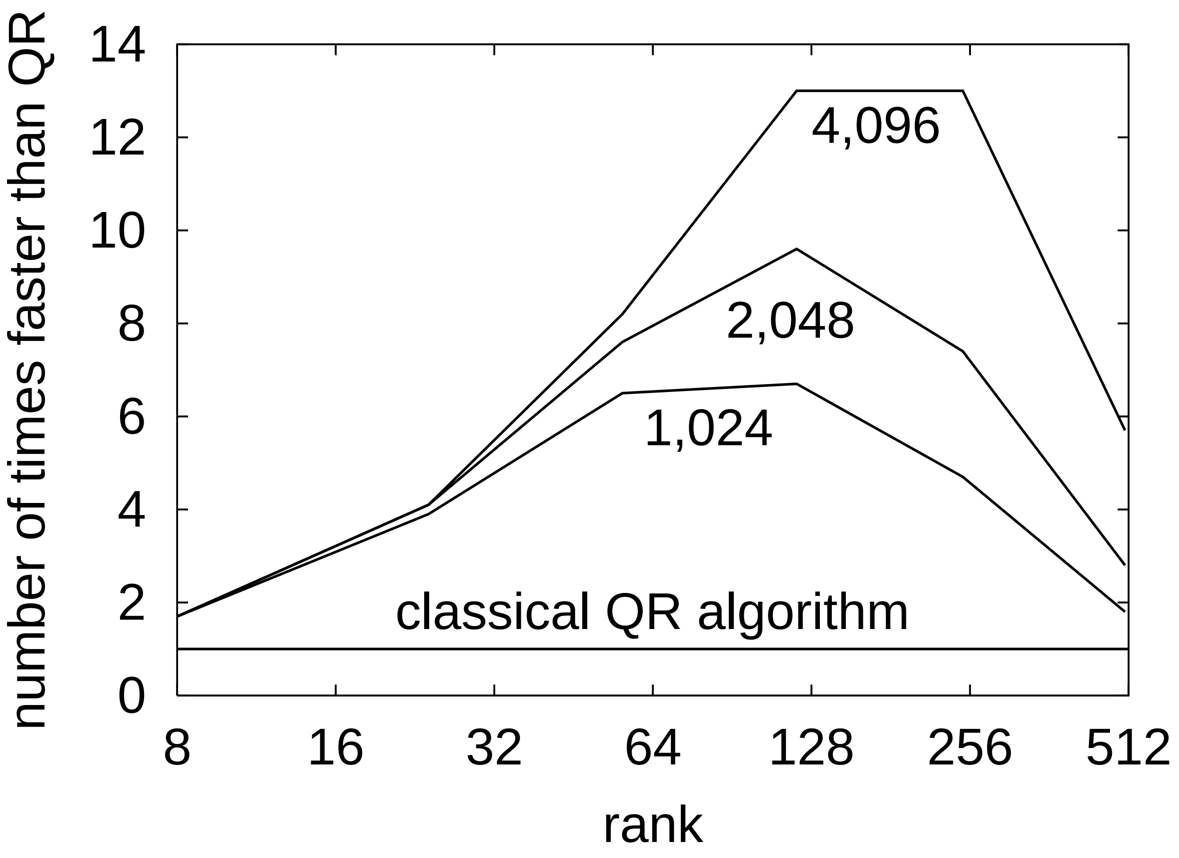
- F is the discrete Fourier transform, $F_{jk} = e^{-2\pi i (j-1)(k-1)/N}$.

- 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.

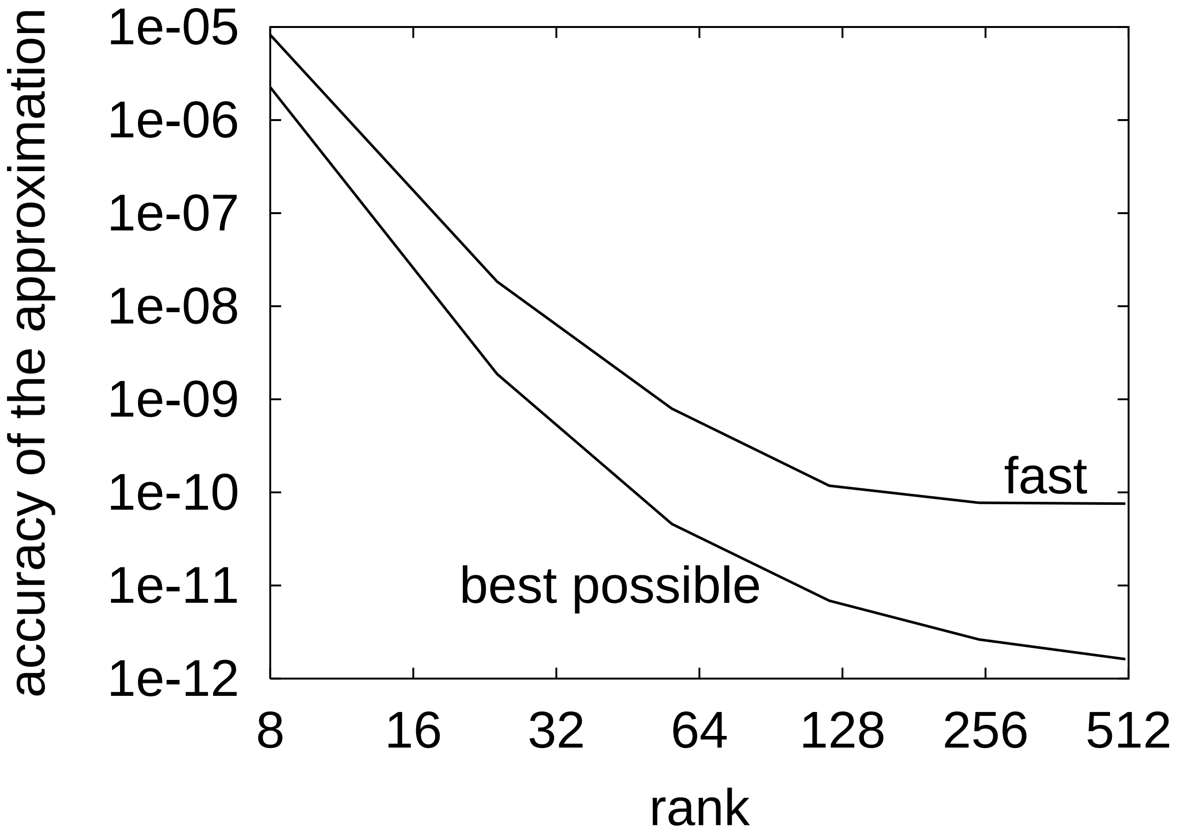
This idea was described by Nir Ailon and Bernard Chazelle (2006).
There is also related recent work by Sarlós (on randomized regression).

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 $N \times N$ matrix.

Let k be an integer, and let ℓ be an integer such that $\ell \geq k$.

Let Ω be an $N \times \ell$ matrix with i.i.d. Gaussian elements.

Let Q be an $N \times \ell$ matrix whose columns form an ON-basis for the columns of $A\,\Omega$.

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

Then

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

with probability at least

$$1 - \varphi(\ell - k),$$

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:

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

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.)

In applications where the singular values do not decay rapidly (as is typical for statistical data analysis, analysis of networks, *etc*), better estimates are needed.

Reducing the factor is a topic of active research — but it appears that in most environments, the factor can more or less entirely be eliminated.

- "A posteriori" analysis — Halko/Martinsson/Tropp — Sep. 2008.
- Use of power iterations — Rokhlin/Szlam/Tygert — Oct. 2008.
- Improved proofs — Halko/Martinsson/Tropp — Dec. 2008.
- That the "fixes" work has been demonstrated by examples with $N \sim 10^7$ and $\sigma_{k+1} \sim 10^{-2}$ — Martinsson/Tygert/Shkolnisky — Dec. 2008.

Notes on the proof:

We assume without loss of generality that A is diagonal and square since:

- The probability distribution of a Gaussian matrix is invariant under rotations.
- All steps of the algorithm are also invariant under rotations.

Then

$$A\,\Omega = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \sigma_N \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1\ell} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2\ell} \\ \vdots & \vdots & & \vdots \\ \omega_{N1} & \omega_{N2} & \cdots & \omega_{N\ell} \end{bmatrix}.$$

The question becomes: How large must ℓ be before we can say with high probability that the columns of an $N \times \ell$ matrix populated with i.i.d. Gaussian random variables with high probability sample the first k coordinates?

At this point, there is a rich literature in probability theory to draw on. The arguments are based on "concentration of mass" and the key insight is that "thin" random matrix tend to be well-conditioned (or at least have well-conditioned sub-matrices) and sample high-dimensional space with almost optimal efficiency.

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*

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

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

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

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, Tau, Romberg, Donoho).
- Geometry of point clouds in high dimensions (Coifman, Maggioni, Jones, *etc*).
- Construction of multi-resolution SVDs.
- Clustering algorithms.
- Search algorithms / knowledge extraction.
- Approximate nearest neighbor search.

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 among 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 tend to be very focussed 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

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

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 $O(N^2\,(\log N)^2)$ 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.

We are currently working on a number of applications:

- Acceleration of classical linear algebra routines.

- Construction of reduced models for physical phenomena (“model reduction”).
  - Modeling of percolating micro-structures.
  - Wave propagation through media with periodic micro-structures.
  - Crack propagation through composite materials.
  - Scattering problems involving multiple scatterers.

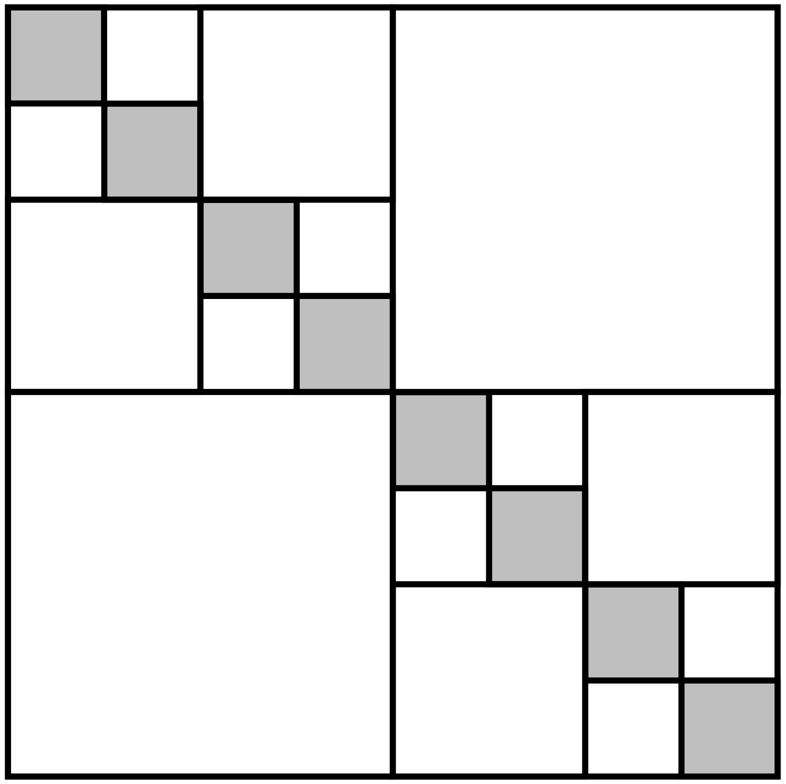
- Statistical data analysis / “principal component analysis”:
  - Genomics.
  - Image processing.

- Analysis of network matrices (such as ones representing the link structure of the World Wide Web) — knowledge extraction

- Matrices whose *off-diagonal blocks* are rank deficient.

We say that a matrix A is "block rank-deficient" if its off-diagonal blocks have limited numerical rank.

The following figure shows one of the most common tessellations:



Block rank-deficient matrices are extremely common in scientific computing.

Example — Particle simulations:

Let $\{x_i\}_{i=1}^N$ be a number of points in space, $x_i \in \mathbb{R}^3$.

Let $\{q_i\}_{i=1}^N$ be the strength of "electric charges" placed at the points x_i.

Letting u_i denote the electric potential at point x_i, we have

$$u_i = \sum_{j \neq i} \frac{1}{4\,\pi\,|x_i - x_j|}\,q_j = [A\,q]_i,$$

where the matrix A is defined via

$$A_{ij} = \begin{cases} \frac{1}{4\,\pi\,|x_i - x_j|}, & i \neq j, \\ 0 & i = j. \end{cases}$$

This matrix A is "block rank-deficient".

This observation underlies the so called "Fast Multipole Method".

Applications in astrophysical simulation, biochemistry, modeling of semi-conductors, *etc.*

... Shiv’s picture ...

Example — Integral equation methods for elliptic PDEs:

Integral operators (single layers, double layers, *etc*) associated with PDEs such as

- Laplace's equation,
- Helmholtz equation (at low and intermediate frequencies),
- the equations of elasticity,
- the Yukawa equation,
- *etc*

turn into "block rank-deficient" matrices upon discretization.

Discretizations of Green's functions on bounded domains ...

Approximations of Dirichlet-to-Neumann operators ...

Etc, etc, ...

Block rank-deficient matrices abound in this environment.

A fast direct solver for boundary integral operators
(Martinsson & Rokhlin, 2004)

Computational results for the double layer potential associated with an exterior Laplace problem on a smooth contour:

Relative accuracy: 10^{-10}

Problem size: $102\,400 \times 102\,400$.

Time for initial inversion: 19 seconds.

Time for applying the inverse: 0.17 seconds.

Computational results for the combined field equation associated with an exterior Helmholtz problem on a smooth contour 200 wave-lengths in size:

Relative accuracy: 10^{-8}

Problem size: $25\,600 \times 25\,600$.

Time for initial inversion: 7 minutes.

Time for applying the inverse: 0.79 seconds.

Note: This method is not "fast" at very large wave numbers.

Example — Matrices approximating elliptic differential operators:

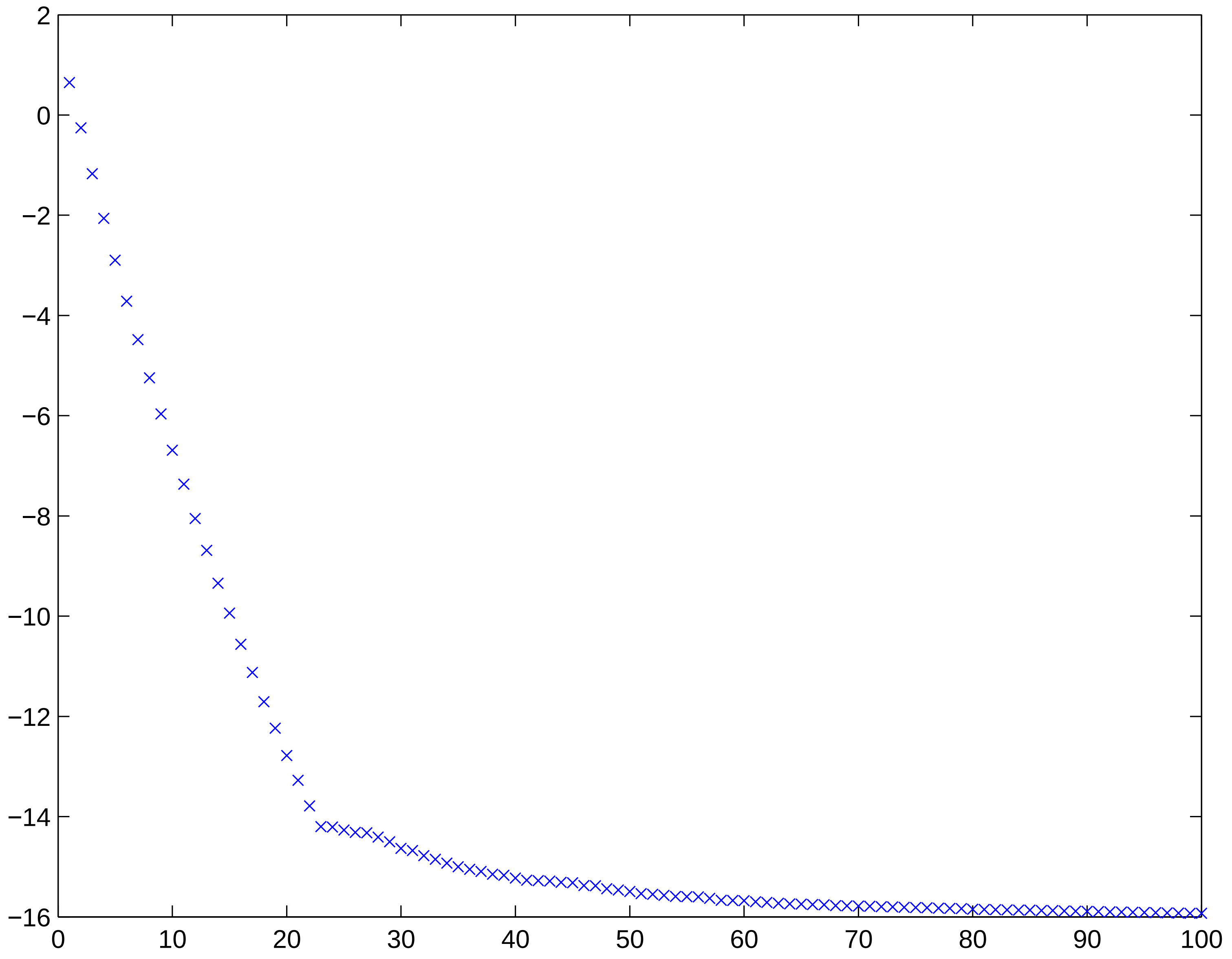
Let L be the standard five-point stencil (discrete Laplacian) on a 50×50 grid:

$$L = \begin{bmatrix} C & -I & 0 & 0 & \cdots \\ -I & C & -I & 0 & \cdots \\ 0 & -I & C & -I & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \qquad C = \begin{bmatrix} 4 & -1 & 0 & 0 & \cdots \\ -1 & 4 & -1 & 0 & \cdots \\ 0 & -1 & 4 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}.$$

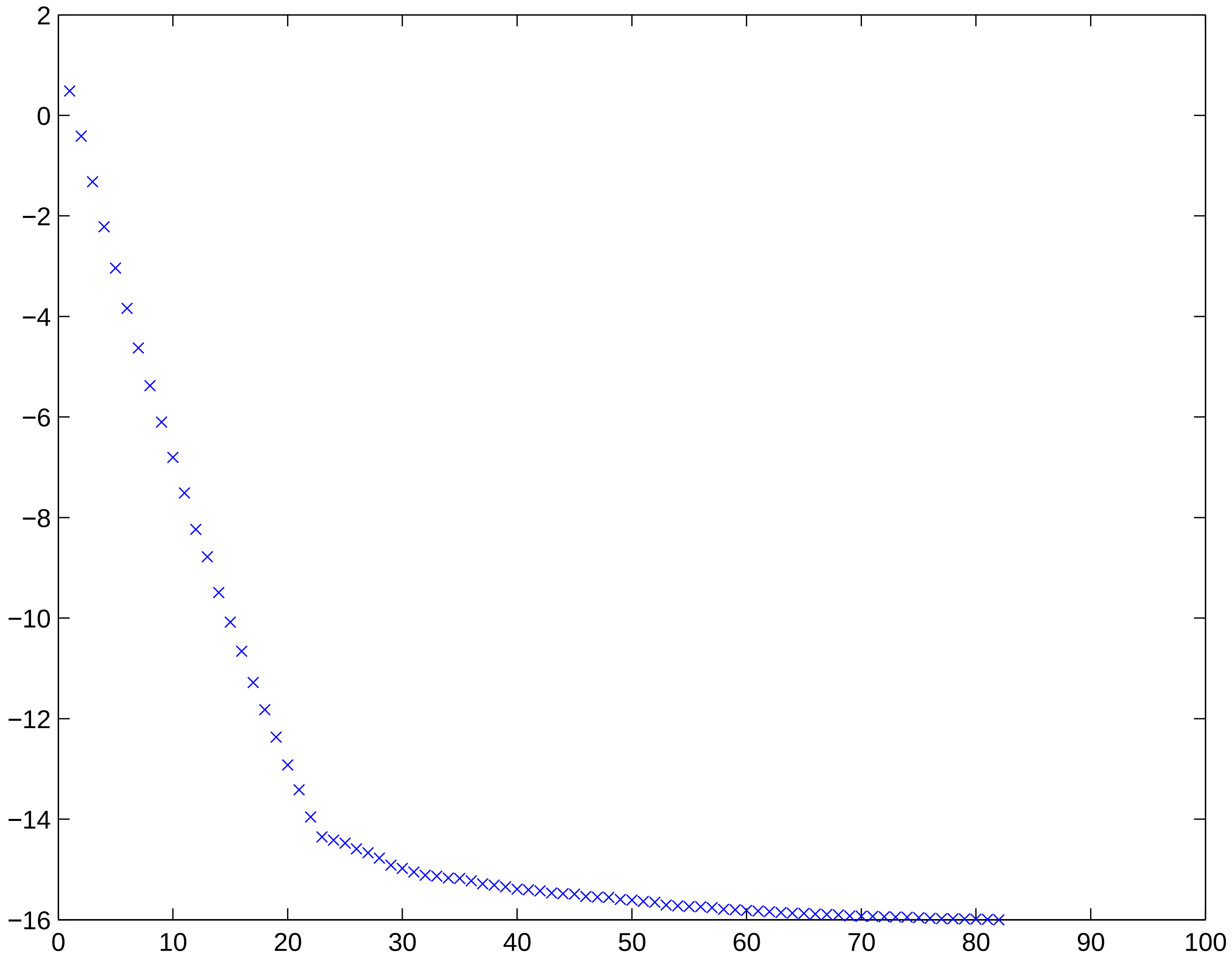
Let A be the inverse of L, and partition it:

$$A = L^{-1} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}.$$

We consider the 625×625 submatrix A_{14} of the $2\,500 \times 2\,500$ matrix A.



The 10-logarithm of the singular values of A_{14}.



The 10-logarithm of the singular values of A_{14} — now with *random coefficients*.

A fast direct solver for finite element matrices
(Martinsson, 2007)

We have constructed an $O(N\,(\log N)^2)$ scheme for directly inverting the matrix L.

For $N = 1\,000\,000$, the scheme requires 4 minutes for the initial inversion (on an 2.8 GHz Pentium IV desktop with 512 Mb of RAM) performed at single precision (seven correct digits).

Subsequent problems that are loaded on the boundary only can be solved in 0.03 seconds (provided that only the solution on the boundary is sought).

The scheme is very primitive. Work that will improve the asymptotic time requirement to $O(N\,\log N)$ (or possibly $O(N)$) is under way. The constants will be much smaller.

Works for a wide range of elliptic equations.

Similar results have recently been obtained by Gu, Chandrasekaran, Xia, Li.

Example — Toeplitz matrices:

Let $\dots, a_{-2}, a_{-1}, a_0, a_1, a_2, \dots$ be complex numbers.

For a positive integer N, let A denote the $N \times N$ matrix with entries

$$A_{ij} = a_{j-i}.$$

For instance, for $N = 4$, we get

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_{-1} & a_0 & a_1 & a_2 \\ a_{-2} & a_{-1} & a_0 & a_1 \\ a_{-3} & a_{-2} & a_{-1} & a_0 \end{bmatrix}.$$

Then the Fourier transform of A,

$$\hat{A} = F_N \, A \, F_N^*$$

is a “block rank-deficient” matrix. (This is not necessarily true for A itself.)

Again, such matrices can be inverted rapidly. (Martinsson, Rokhlin, Tygert, 2005)

Example — Sampling of "sinc" matrix:

Let $\{x_j\}_{j=1}^{N}$ denote real numbers such that $-\infty < x_1 < x_2 < \cdots < x_{N-1} < x_N < \infty$.

Define for a real positive number c the $N \times N$ matrix A by

$$A_{ij} = \begin{cases} c & i = j, \\ \dfrac{\sin\bigl(c\,(x_i - x_j)\bigr)}{x_i - x_j}, & i \neq j. \end{cases}$$

Then A is a structured matrix.

Prolate spheroidals are eigenfunctions of the sinc operator — connection to band-filtering signals / image processing / customized Fourier transforms / *etc.* *(Work in progress.)*

To summarize:

“Block rank-deficient” matrices are ubiquitous in scientific computing:

- Numerical methods for elliptic and parabolic PDEs.
- Models of N particles with linear interactions (*e.g.* astrophysical simulations).
- Signal processing (with matrices representing transforms and filters).
- Matrices that model networks are typically *sparse* — factorizing them often leads to dense but “block rank-deficient” matrices.

The structure of “block rank-deficient” can be exploited to construct accelerated — $O(N)$ or $O(N\,(\log N)^p)$ — algorithms for standard operations:

- Matrix-vector multiply. (“Fast Multipole Method”, “Barnes-Hut”, *etc.*)
- [New!] Matrix inversion.
- [New!] Matrix-matrix multiplication.
- [New!] Matrix factorization (QR, LU, Cholesky, *etc*).
- [Hopefully upcoming. . .] Partial eigenvalue decompositions, SVDs.

Existing fast algorithms for performing matrix-vector multiplies for block rank deficient matrices typically exploit known analytic properties of the off-diagonal blocks (they are oftentimes samples of a known kernel function).

A key difficulty in constructing matrix inversion schemes is that such precise knowledge about the kernel of the inverse functions is typically not available.

This is where the randomized schemes come in!

Observation [Martinsson 2008]: The randomized sampling techniques can be used to invert certain "block rank-deficient" matrices given only:

- A "black box" fast matrix-vector multiplier (such as a legacy code).
- A subroutine that computes individual entries of the matrix.

Summary

Randomized methods for approximating rank-deficient matrices.

Randomized methods work for "block rank-deficient" matrices, and thus enable the development of $O(N)$ methods for a wide range of matrix computations.

The methods presented are closely related to recent developments in functional analysis and random matrix theory. The methods described are in fact much simpler than other recent work. However, they appear to have been overlooked.

The randomized methods have resolved a number of long-standing problems in numerical analysis, for instance:

- Definitively overcome issues relating to stability and parallelization of Lanzcos.
- Fast compression and inversion of "block rank-deficient" matrices.
- Reduction of CPU time requirement of partial spectral decompositions, QR factorizations, *etc*, from $O(N^2\,k)$ to $O(N^2\log(k))$.
- Computing SVDs / PCAs of huge matrices with low signal-to-noise ratios.

There should be much much more to come.