Randomized methods for the approximation of matrices

Gunnar Martinsson, The University of Colorado at Boulder

Acknowledgements:

Some of the material presented is joint work with Vladimir Rokhlin and Mark Tygert.

Some of the material presented is work by Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert

Notation — Singular Value Decompositions:

Let A be an $m \times n$ matrix, where $m > n$. Then we write the SVD of A as

$$\underbrace{A}_{m\times n} = \underbrace{U}_{m\times n}\,\underbrace{D}_{n\times n}\,\underbrace{V^{\mathrm{t}}}_{n\times n},$$

where U and V are matrices whose columns are orthonormal, and

$$D = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_n \end{bmatrix}.$$

Following standard practise, we order the singular values σ_j so that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0,$$

and set

$$\sigma_1 := \sigma_{\max} = ||A||,$$

$$\sigma_n := \sigma_{\min}.$$

Notation — "random-direction" vectors:

Let ω be a vector in $\mathbb{R}^n$ whose every entry is an i.i.d. random variable drawn from the normalized Gaussian distribution, $N(0, 1)$.

Set

$$\tilde{\omega} = \frac{\omega}{||\omega||}.$$

Then $\tilde{\omega}$ is a vector drawn from a uniform distribution on the surface of the unit ball. We call such a vector a "random-direction" vector.

Random sampling — a model problem:

Consider the following situation:

- We seek to determine a bound for the spectral norm $||A||$ of an operator A.
- We have a fast algorithm for matrix-vector multiplication: $x \mapsto A\,x$.

The basic idea is to generate a set of random vectors,

$$\{\tilde{\omega}_1,\ \tilde{\omega}_2,\ \ldots\ ,\ \tilde{\omega}_q\},$$

and then determine a bound for $||A||$ from the set

$$\{A\,\tilde{\omega}_1,\ A\,\tilde{\omega}_2,\ \ldots\ ,\ A\,\tilde{\omega}_q\}.$$

(We will talk about more challenging problems than estimating $||A||$ later.)

One extreme situation: A is well-conditioned.

Let κ denote the condition number of A, and let $\tilde{\omega}$ be a random-direction vector, then

$$||A\,\tilde{\omega}|| \geq \sigma_{\min} = \frac{\sigma_{\min}}{\sigma_{\max}}\,||A|| = \frac{1}{\kappa}\,||A||.$$

In this case, an assured bound for $||A||$ is obtained from a single evaluation of $||A\,\tilde{\omega}||$:

$$||A|| \leq \kappa\,||A\,\tilde{\omega}||.$$

The opposite extreme: A has rank one.

Suppose that $A = \sigma_{\max}\,u\,v^{\mathrm{t}}$, then

$$||A\,\tilde{\omega}|| = ||\sigma_{\max}\,u\,(v\cdot\tilde{\omega})|| = \sigma_{\max}\,||u||\,|v\cdot\tilde{\omega}| = ||A||\,|v\cdot\tilde{\omega}|.$$

Letting θ denote the angle between v and $\tilde{\omega}$ we have $v\cdot\tilde{\omega} = \cos\theta$ and so

$$||A|| = \frac{1}{|v\cdot\tilde{\omega}|}\,||A\,\tilde{\omega}|| = \frac{1}{|\cos\theta|}\,||A\,\tilde{\omega}||.$$

The generic situation.

Recall the Singular Value Decomposition of A

$$\begin{array}{ccccc} A & = & U & D & V^{\mathrm{t}} \\ m \times n & & m \times n & n \times n & n \times n \\ & & \text{ON columns} & \text{diagonal} & \text{unitary} \end{array}$$

Let $\tilde{\omega}$ be a random-direction vector. Then

$$||A\,\tilde{\omega}|| = ||U\,D\,V^{\mathrm{t}}\,\tilde{\omega}|| = \{\text{Set } \tilde{\nu} = V^{\mathrm{t}}\,\tilde{\omega}\} = ||U\,D\,\tilde{\nu}|| = ||D\,\tilde{\nu}|| = \left(\frac{\sum_{j=1}^{n}(\sigma_j\,\nu_j)^2}{\sum_{j=1}^{n}\nu_j^2}\right)^{1/2}.$$

Note that $\tilde{\nu}$ has the same distribution as $\tilde{\omega}$ since V is unitary. Then

$$||A\,\tilde{\omega}|| \geq \frac{\sigma_1\,|\nu_1|}{\sqrt{\sum_{j=1}^{n}\nu_j^2}} = ||A||\,\frac{|\nu_1|}{\sqrt{\sum_{j=1}^{n}\nu_j^2}} \approx ||A||\,\frac{|\nu_1|}{\sqrt{n}}.$$

The worst case scenario is when A has rank 1 — in this case the first inequality is in fact an equality.

Recall: $||A\,\tilde{\omega}|| \geq ||A||\,\dfrac{|\nu_1|}{\sqrt{n}}$ where $\nu_1 \in N(0,1)$.

Fix any $\mu \in (0,1)$, then

$$\mathbf{P}\left(||A|| \geq \frac{1}{\mu}\,||A\,\tilde{\omega}||\right) \leq \mathbf{P}\left(||A|| \geq \frac{1}{\mu}\,||A||\,\frac{|\nu_1|}{\sqrt{n}}\right) = \mathbf{P}\left(\mu\sqrt{n} \geq |\nu_1|\right)$$

$$= \int_{-\mu\sqrt{n}}^{\mu\sqrt{n}} \frac{e^{-x^2}}{\sqrt{2\pi}}\,dx \leq \frac{1}{\sqrt{2\pi}}\,2\mu\sqrt{n} \approx 0.8\,\mu\,\sqrt{n}.$$

One option is to set $\mu = 1/(10\,\sqrt{n})$. Then

$$10\,\sqrt{n}\,||A\,\tilde{\omega}||$$

is an upper bound for $||A||$ with confidence 0.9. Repeating the experiment 10 times, we'd find that

$$10\,\sqrt{n}\,\max(||A\,\tilde{\omega}_1||,\,\ldots\,,\,||A\,\tilde{\omega}_{10}||)$$

is an upper bound for $||A||$ with confidence $1 - 10^{-10}$.

A better option: Apply power iteration to a single random vector $\tilde{\omega}$.

Recall the estimate

$$\mathbf{P}\left(||A|| \geq \frac{1}{\mu}\,||A\,\tilde{\omega}||\right) \leq 0.8\,\mu\,\sqrt{n}.$$

Replacing A by $(A^*\,A)^k$ and setting $\mu = 10^{-2k}$ we obtain

$$\mathbf{P}\left(||(A^*\,A)^k|| \geq 10^{2k}\,||(A^*\,A)^k\,\tilde{\omega}||\right) \leq 0.8 \cdot 10^{-2k}\,\sqrt{n}.$$

Since $||(A^*, A)^k|| = ||A||^{2k}$ it follows that

$$\mathbf{P}\left(||A|| \geq 10\,||(A^*\,A)^k\,\tilde{\omega}||^{1/2k}\right) \leq 0.8 \cdot 10^{-2k}\,\sqrt{n}.$$

Two observations:

1. The principal advantage of a randomized scheme is that it "explores", or "senses", all directions at once.

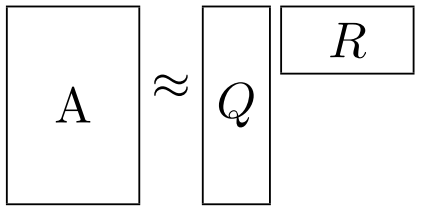
   The higher the dimension n, the thinner the random vector gets spread across the various coordinate directions. However, the weight on each direction only decreases as $1/\sqrt{n}$.

   The $1/\sqrt{n}$ factor can hurt us if n is large and the accuracy in the matrix-vector multiply is low.

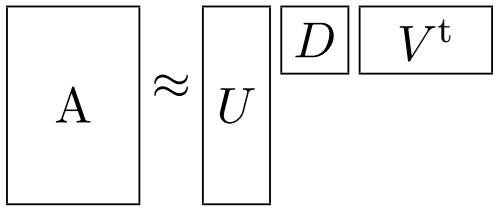
2. Various choices of random vectors are possible, but the "random-direction" distribution works very well. Moreover, the isotropy of this distribution is very convenient in the analysis. In effect, it often allows us to treat the matrix being sampled as a diagonal matrix.

Next we consider a more interesting problem.

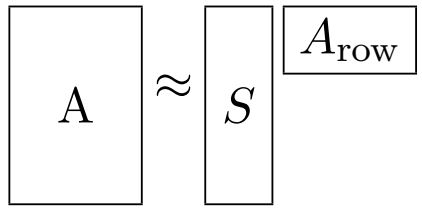
Let A be an $m \times n$ matrix that can be approximated by a matrix of rank k:



"QR-decomposition"



"SVD"



"Interpolative decomposition"

Question: How do you efficiently find such approximations?

A classical answer: Compute an ON-basis for the columns of A using *e.g.* Gram-Schmidt. Cost is $O(mnk)$.

Algorithm 1: When matrix-vector products $x \mapsto A\,x$ can be computed cheaply, say at a cost T_{mult}, the total cost can be reduced to $O(T_{\text{mult}}\,k + m\,k^2)$.

Algorithm 2: When A is a general matrix (not necessarily cheap to apply), the cost can be reduced to $O(m\,n\,\log(k) + (m+n)\,k^2)$.

Algorithms 1 and 2 are based on randomized sampling, meaning that they have a probability of failure. This probability is typically negligible (like 10^{-17}).

Note:
The output of the algorithms is *an orthonormal basis for the column space of* A.

From this basis, any factorization can be computed.

Say that Q is an $m \times k$ matrix with ON-columns such that:

$$A = Q\, Q^{\mathrm{t}}\, A.$$

Then compute $Q^{\mathrm{t}}\, A$ and then compute the SVD of this $k \times n$ matrix:

$$Q^{\mathrm{t}}\, A = \tilde{U}\, D\, V^{\mathrm{t}}.$$

Then

$$A = Q\, (Q^{\mathrm{t}}\, A) = \underbrace{Q\, \tilde{U}}_{=:U}\, D\, V^{\mathrm{t}} = U\, D\, V^{\mathrm{t}}.$$

In many environments, it is not even necessary to compute $Q^{\mathrm{t}}\, A$ ...

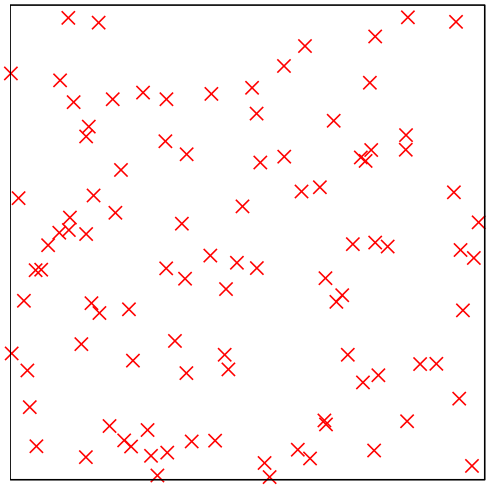
Related work on randomized algorithms:

- Dixon (1983)
- Wozniakowski and Kuczynsky (1993)
- 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)

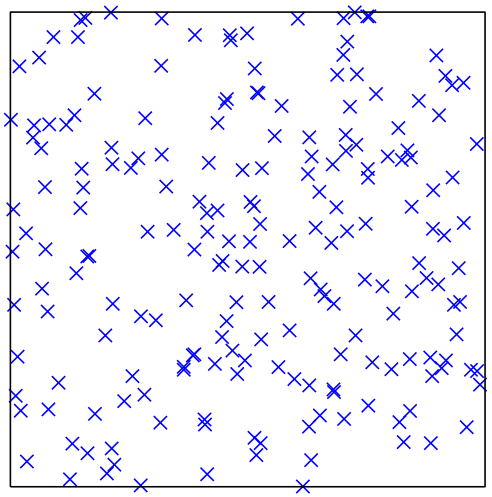
Algorithm 1

For when the matrix-vector multiplication $x \mapsto A\,x$ is cheap.

Example:



Ω_{S}



Ω_{T}

Source points $\{w_j\}_{j=1}^n$ in Ω_{S} at which charges $(q_j)_{j=1}^n$ are given.

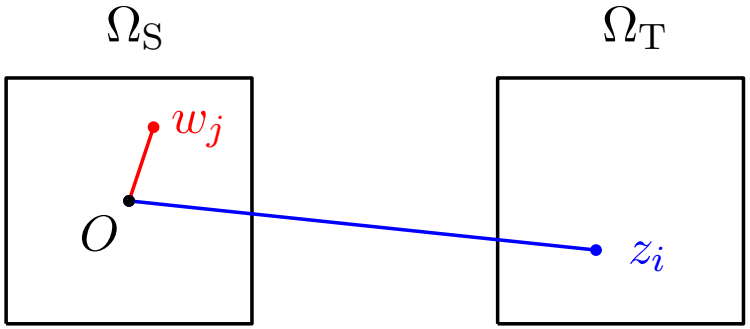
Target points $\{z_i\}_{i=1}^m$ in Ω_{T} at which potentials $(u(z_i))_{i=1}^m$ are sought.

Let A be the $m \times n$ matrix with entries $A_{ij} = \log|z_i - w_j|$. Then

$$u(z_i) = [A\,q]_i = \sum_{j=1}^{n} \underbrace{\log|z_i - w_j|}_{=A_{ij}}\, q_j$$

"A maps a charge distribution to a set of potentials."

Using analysis to obtain an approximate factorization of A:

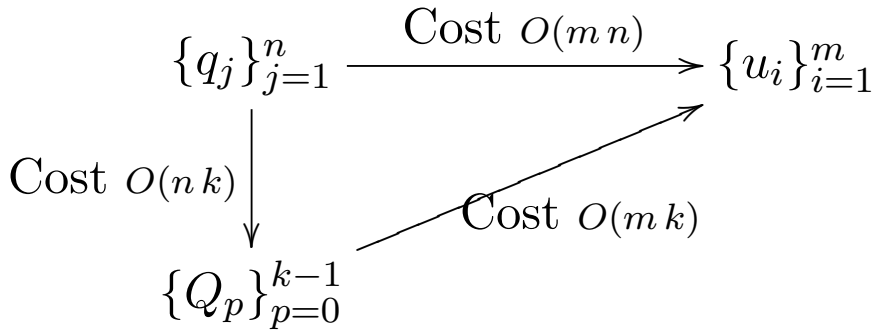


$$
\begin{aligned}
u(z_i) &= \sum_{j=1}^{n} \log(z_i - w_j)\, q_j = \sum_{j=1}^{n} \left[\log z_i + \log(1 - w_j/z_i)\right] q_j \\
&\approx \sum_{j=1}^{n} \left[\log z_i + \sum_{p=1}^{k-1} \frac{-1}{p} \left(\frac{w_j}{z_i}\right)^p\right] q_j \\
&= \log z_i \underbrace{\left(\sum_{j=1}^{n} q_j\right)}_{=:Q_0} + \sum_{p=1}^{k-1} \frac{-1}{p\, z_i^p} \underbrace{\left(\sum_{j=1}^{n} w_j^p\, q_j\right)}_{=:Q_p} \\
&= \log z_i\, Q_0 + \sum_{p=1}^{k-1} \frac{-1}{p\, z_i^p}\, Q_p.
\end{aligned}
$$

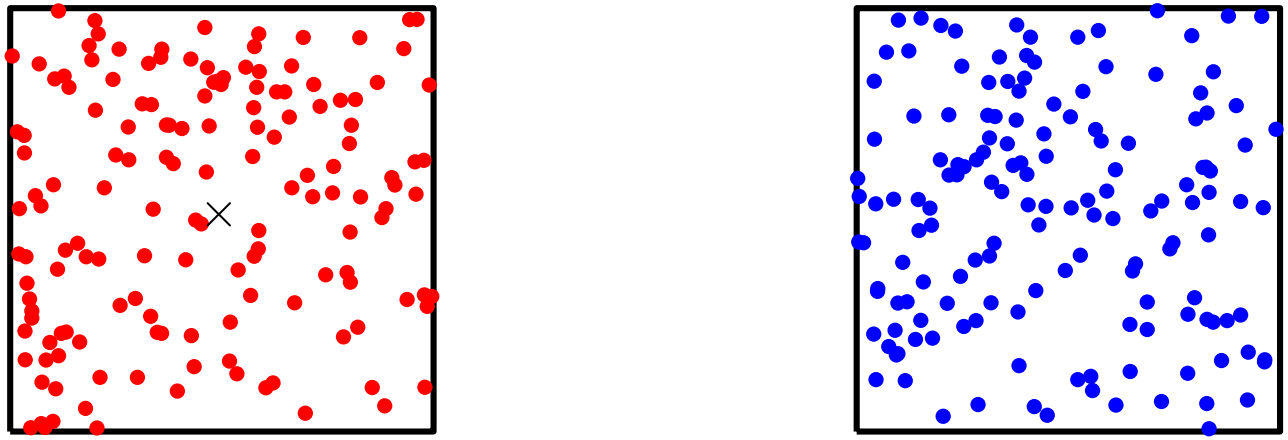
The approximation on the previous page can be written in matrix format as:

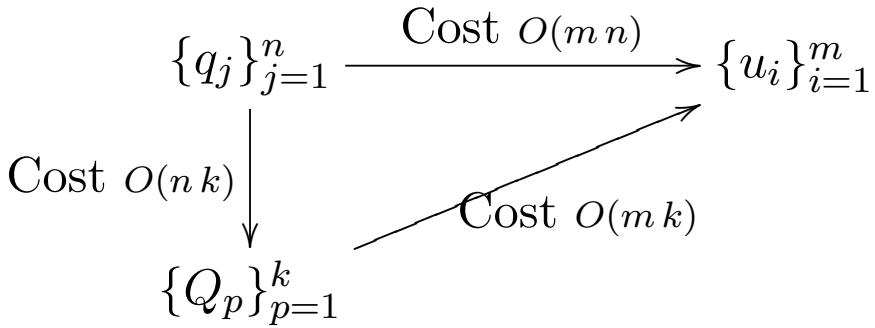
$$\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \approx \begin{bmatrix} \log(z_1) & -\frac{1}{z_1} & -\frac{1}{2\,z_1^2} & \cdots & -\frac{1}{(k-1)\,z_1^{k-1}} \\ \log(z_2) & -\frac{1}{z_2} & -\frac{1}{2\,z_2^2} & \cdots & -\frac{1}{(k-1)\,z_2^{k-1}} \\ \vdots & \vdots & \vdots & & \vdots \\ \log(z_n) & -\frac{1}{z_2} & -\frac{1}{2\,z_n^2} & \cdots & -\frac{1}{(k-1)\,z_n^{k-1}} \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ w_1 & w_2 & \cdots & w_m \\ w_1^2 & w_2^2 & \cdots & w_m^2 \\ \vdots & \vdots & & \vdots \\ w_1^{k-1} & w_2^{k-1} & \cdots & w_m^{k-1} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix}}_{= \begin{bmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_{k-1} \end{bmatrix}}$$

Or, using a diagram ...



We have reduced the computational cost from $O(m\,n)$ to $O(k\,(m+n))$.

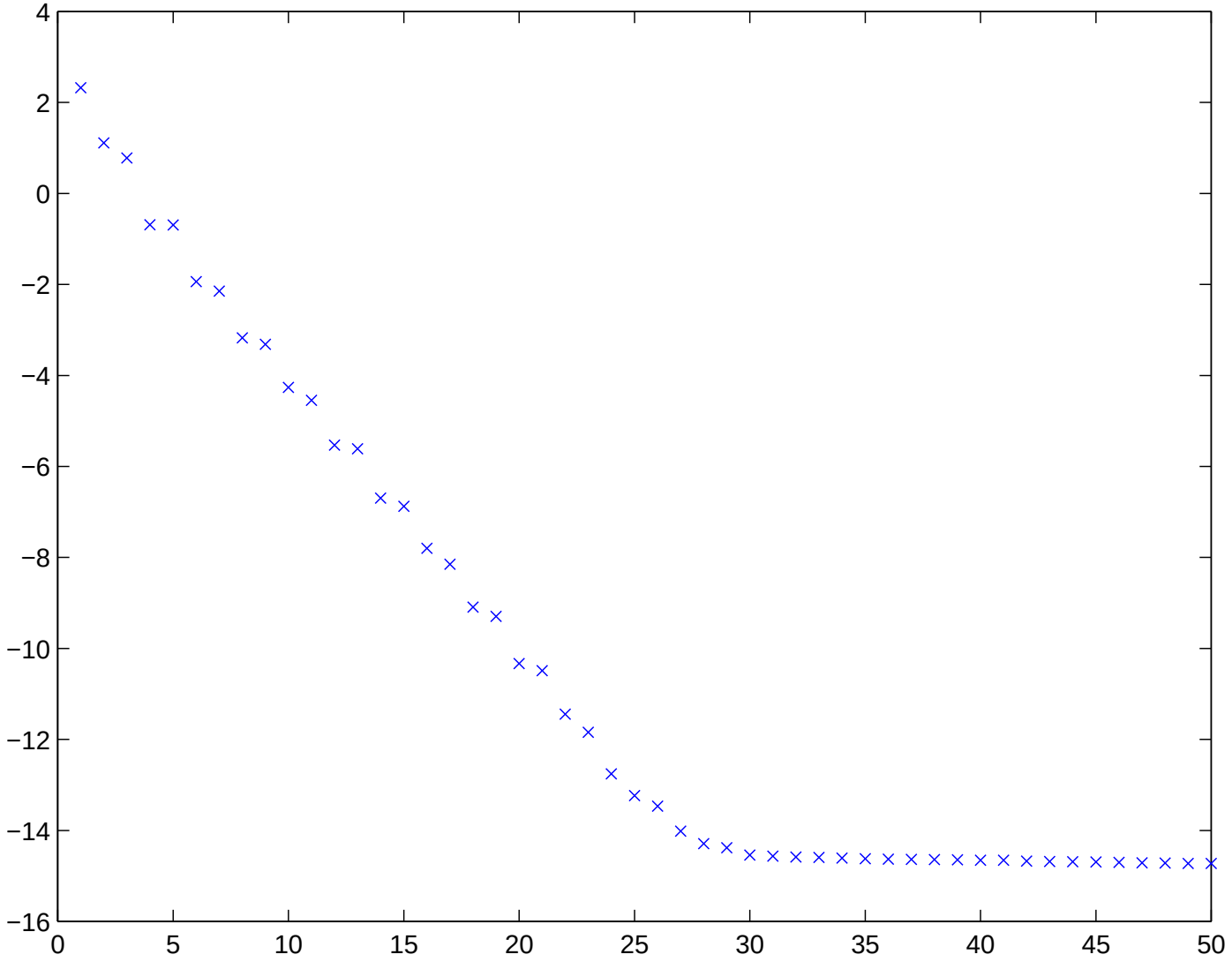




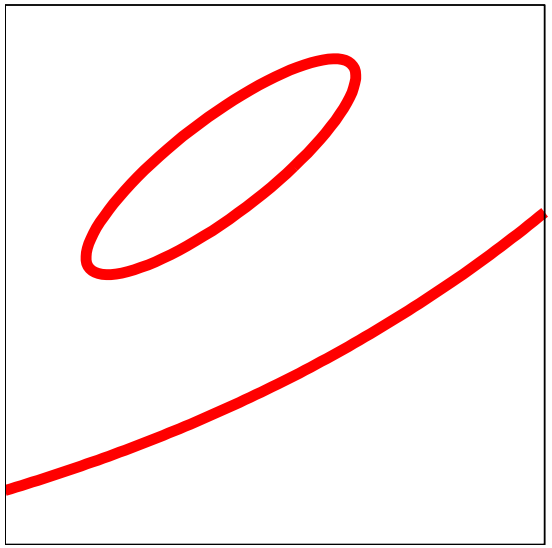
We have reduced the computational cost from $O(m\,n)$ to $O(k\,(m+n))$.

The requested accuracy ε satisfies $\varepsilon \sim \left(\frac{R/\sqrt{2}}{3R/2}\right)^{k+1}$, so $k \sim \frac{\log|\varepsilon|}{\log(3/\sqrt{2})}$.

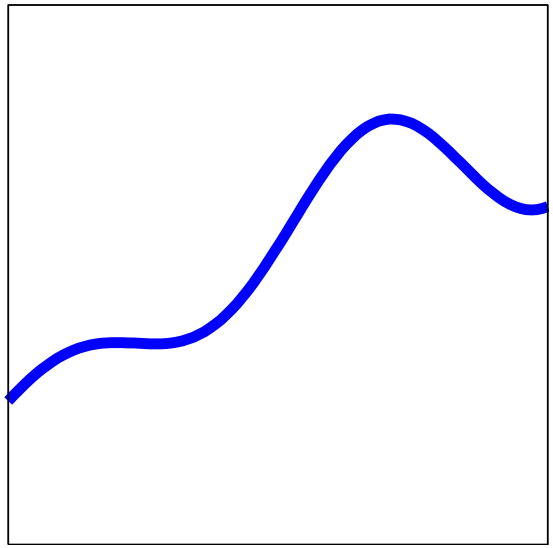
Using the SVD to obtain an approximate factorization of A:



The 10-logarithm of the singular values of A.



Γ_S



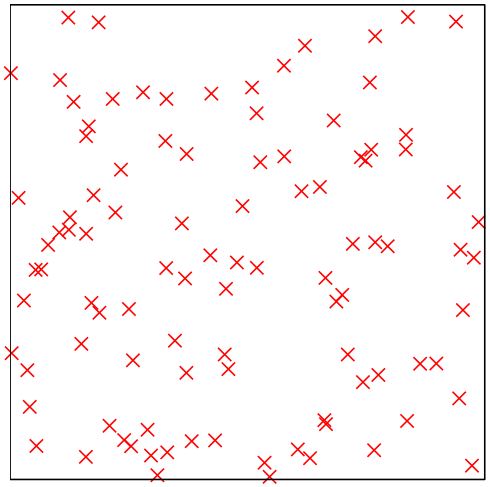
Γ_T

The same type of spectrum is obtained for the "off-diagonal blocks" of many integral operators:

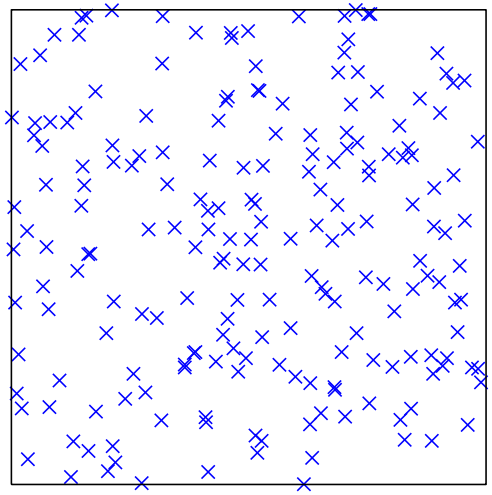
$$[A\,u](x) = \int_{\Gamma_s} G(x,y)\,u(y)\,ds(y), \qquad x \in \Gamma_T.$$

For instance, G could be the single or double layer kernel for the Laplace equation.

Example:



Ω_{S}



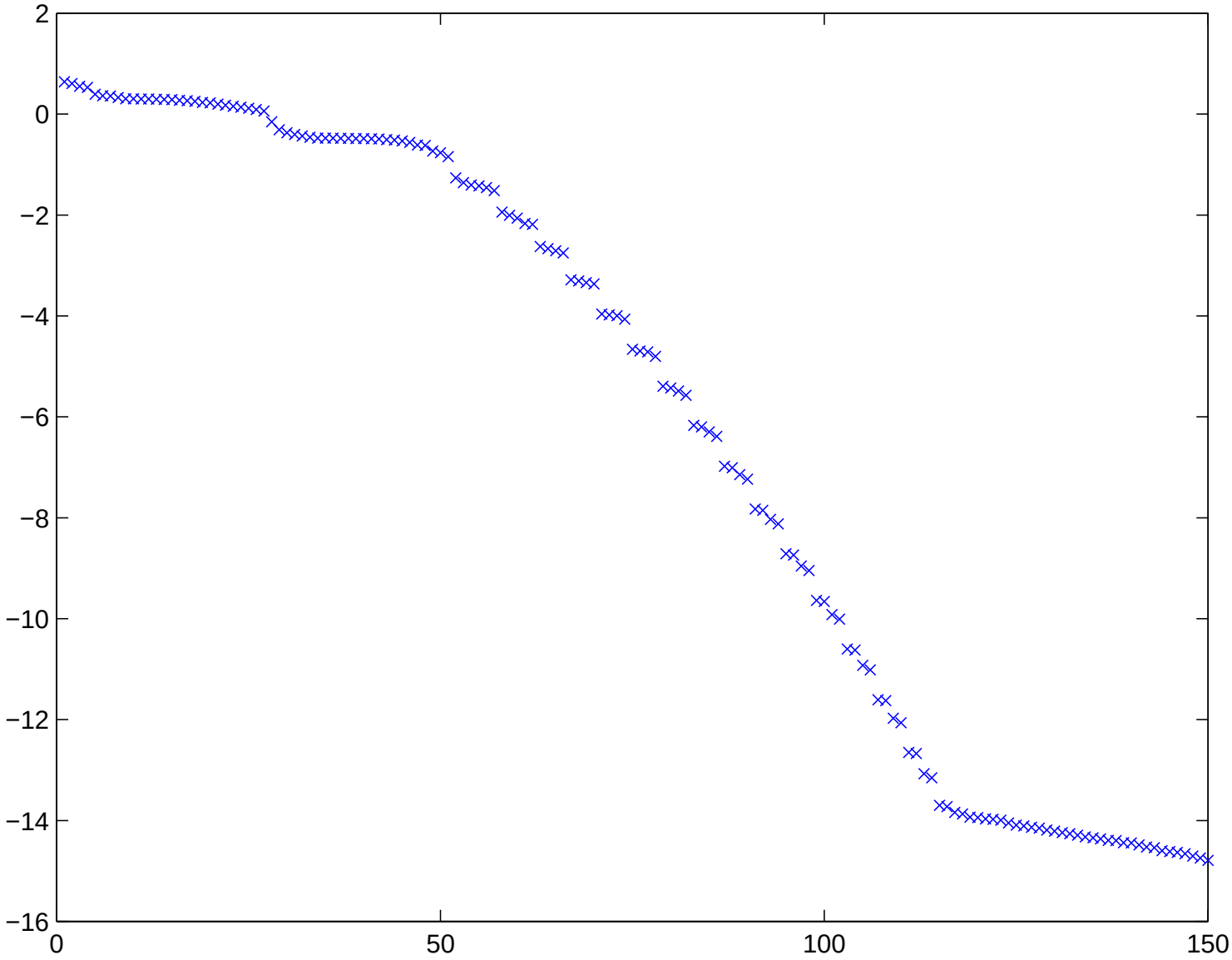
Ω_{T}

Points $\{w_j\}_{j=1}^n$ in Ω_{S} ("sources").

Points $\{z_i\}_{i=1}^m$ in Ω_{T} ("targets").

Let A be an $m \times n$ matrix with entries $A_{ij} = H_0^{(1)}(k\,|z_i - w_j|)$.

"A maps a charge distribution to a set of potentials."



The 10-logarithm of the singular values of A for $k = 35$.

Analysis:	SVD:
• No computation needed to construct the factorization.	• Expensive — costs $O(m\,n\,k)$.
• Suboptimal ranks. Substantially so in many cases.	• Optimal ranks.
• Not always possible or practical.	• Can always be done.

Randomized sampling is cheap, computes factorizations of optimal rank (or very close to it), and always works. (And it doesn't require thinking!)

Example:

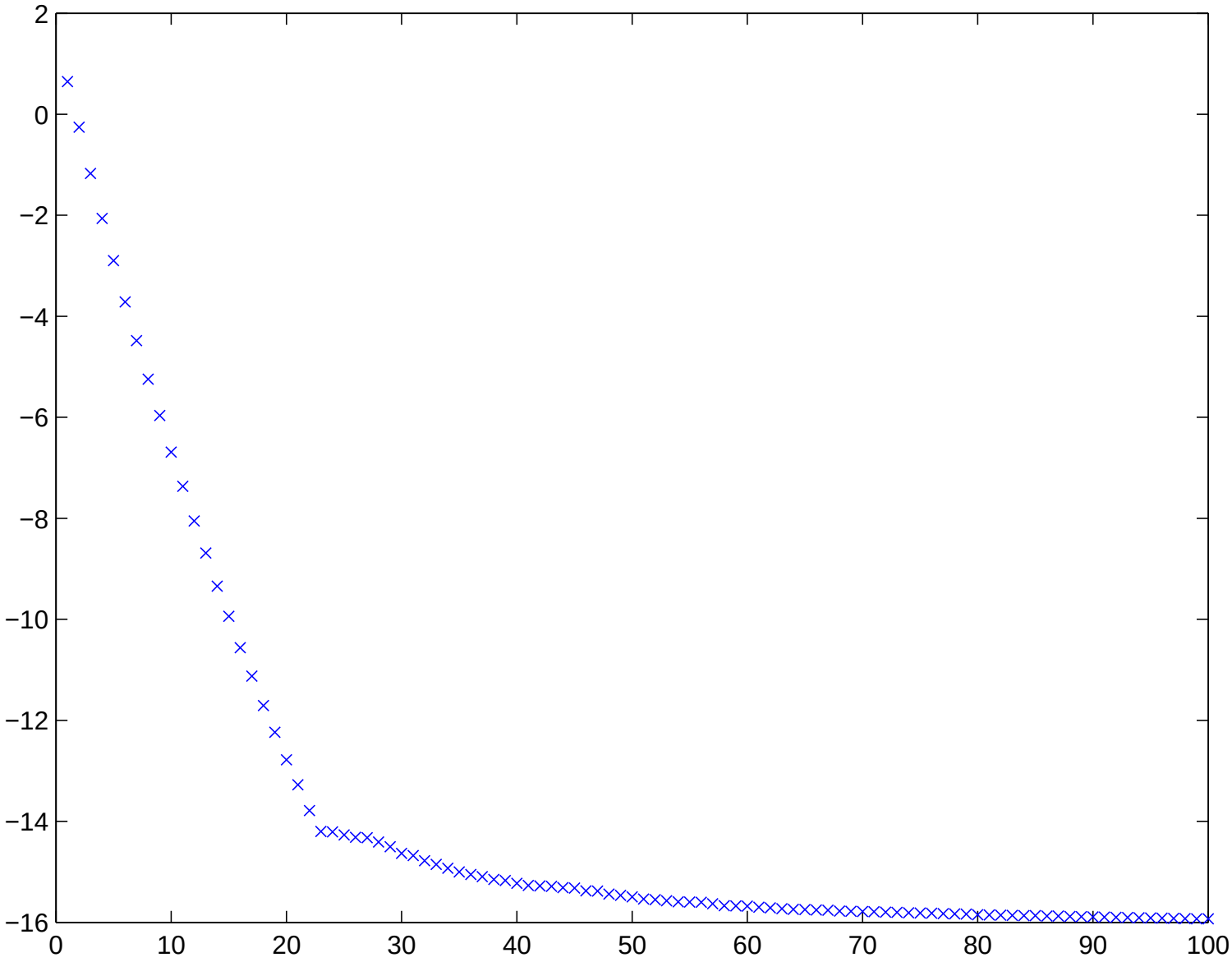
Let L be the standard five-point stencil (discrete Laplacian) on a 50×50 grid:

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

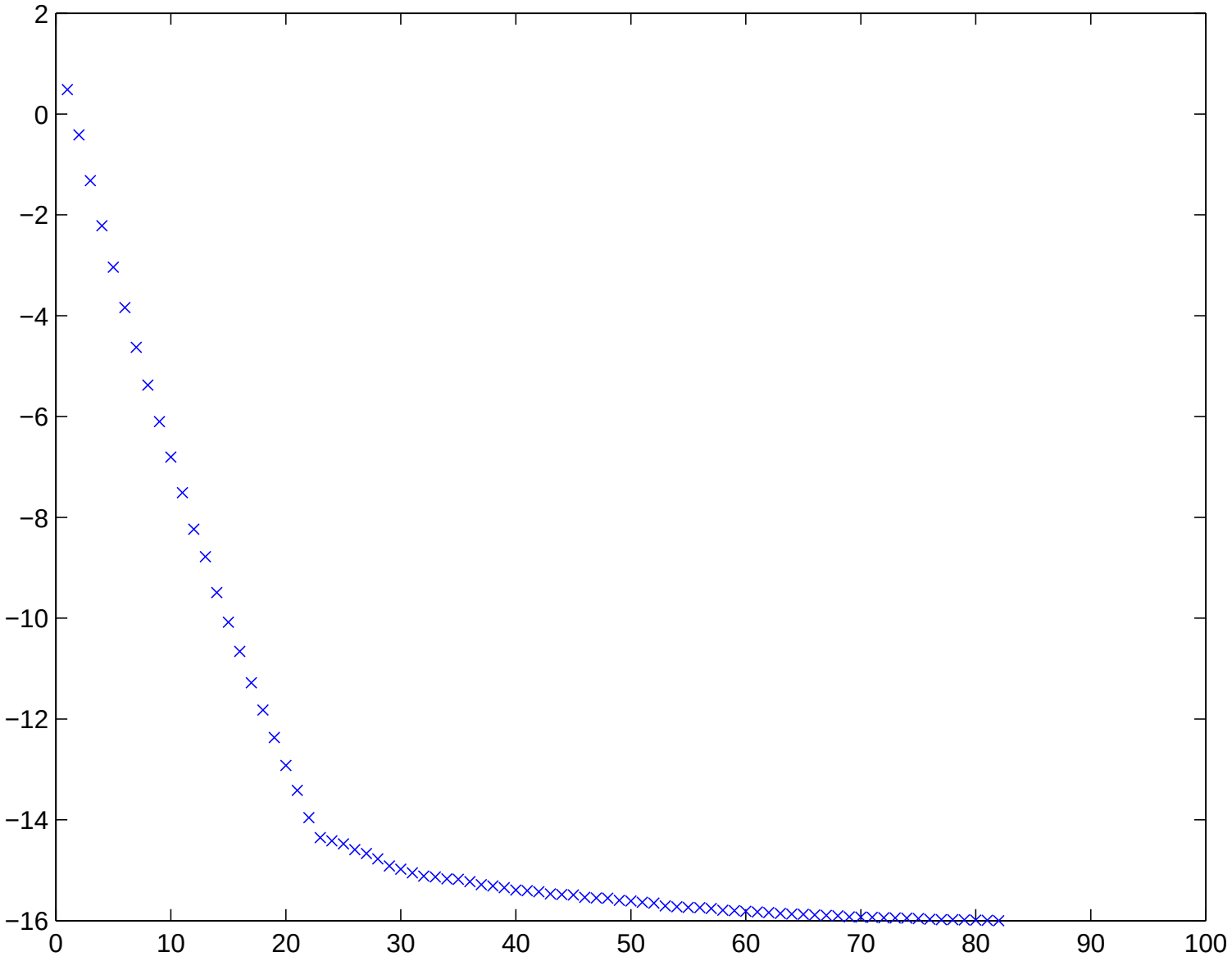
Let A be the inverse of L, and partition it:

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

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*.

Why do we want to compress L^{-1} for a sparse matrix L?

If L results from the discretization of a PDE, then L^{-1} is the direct solution operator. If we can construct L^{-1} rapidly, then we sidestep the need for iterative solvers in solving PDE's rapidly.

If L is a network matrix (representing for instance the World Wide Web), then it is frequently of interest to compute the leading singular vectors of L. An effective method for this is the "shifted inverse power iteration" in which standard power iteration is applied to matrices of the type

$$(L - \mu I)^{-1}$$

(where μ is an estimate of a certain singular value). If we have a compressed representation of L^{-1}, this can often be updated cheaply to obtain the operator $(L - \mu I)^{-1}$.

(Unless additional machinery is constructed and implemented, however, the shifted inverse power method breaks down when μ grows large.)

Algorithm 1 — for when we can compute $x \mapsto A\,x$ rapidly

Recall: A is $m \times n$ with ε-rank k.

Let $\omega_1, \omega_2, \ldots$ be a sequence of random-direction vectors in $\mathbb{R}^n$.

Form[a] the length-m vectors

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

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

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

$$\{y_1, y_2, \ldots, y_l\}$$

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.

---

[a] In some environments, this step involves a sparse linear solve.
If $A = L^{-1}$, then forming $y_1 = A\,\omega_1$ simply means solving $L\,y_1 = \omega_1$.

Algorithm 1 — for when we can compute $x \mapsto A\,x$ rapidly

Recall: A is $m \times n$ with ε-rank k.

Let $\omega_1, \omega_2, \ldots$ be a sequence of random-direction vectors in $\mathbb{R}^n$.

Form[a] the length-m vectors

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

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

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

$$\{y_1, y_2, \ldots, y_l\}$$

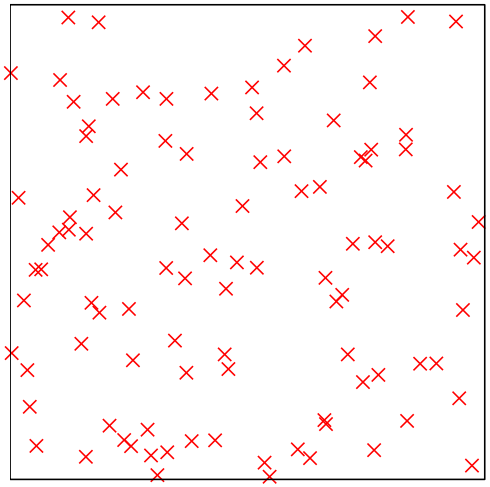
span the column space of A "to within precision ε". Clearly, the probability that this happens gets larger, the larger the gap between l and k.

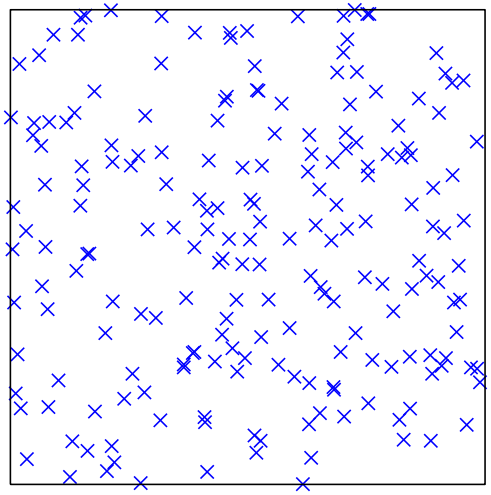
What is remarkable is how fast this probability approaches one.

[a] In some environments, this step involves a sparse linear solve.
If $A = L^{-1}$, then forming $y_1 = A\,\omega_1$ simply means solving $L\,y_1 = \omega_1$.

We illustrate with a numerical example.

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





Generate a sequence $\omega_1, \omega_2, \ldots$ of random vectors in $\mathbb{R}^n$.

Compute $Y_l = [y_1, y_2, \ldots, y_l] = [A\,\omega_1, A\,\omega_2, \ldots, A\,\omega_l]$.

Compute the (column pivoted) QR-factorization $Y_l = Q_l\,R_l\,P_l$.

The "error" after l steps is (using the l^2-operator norm)

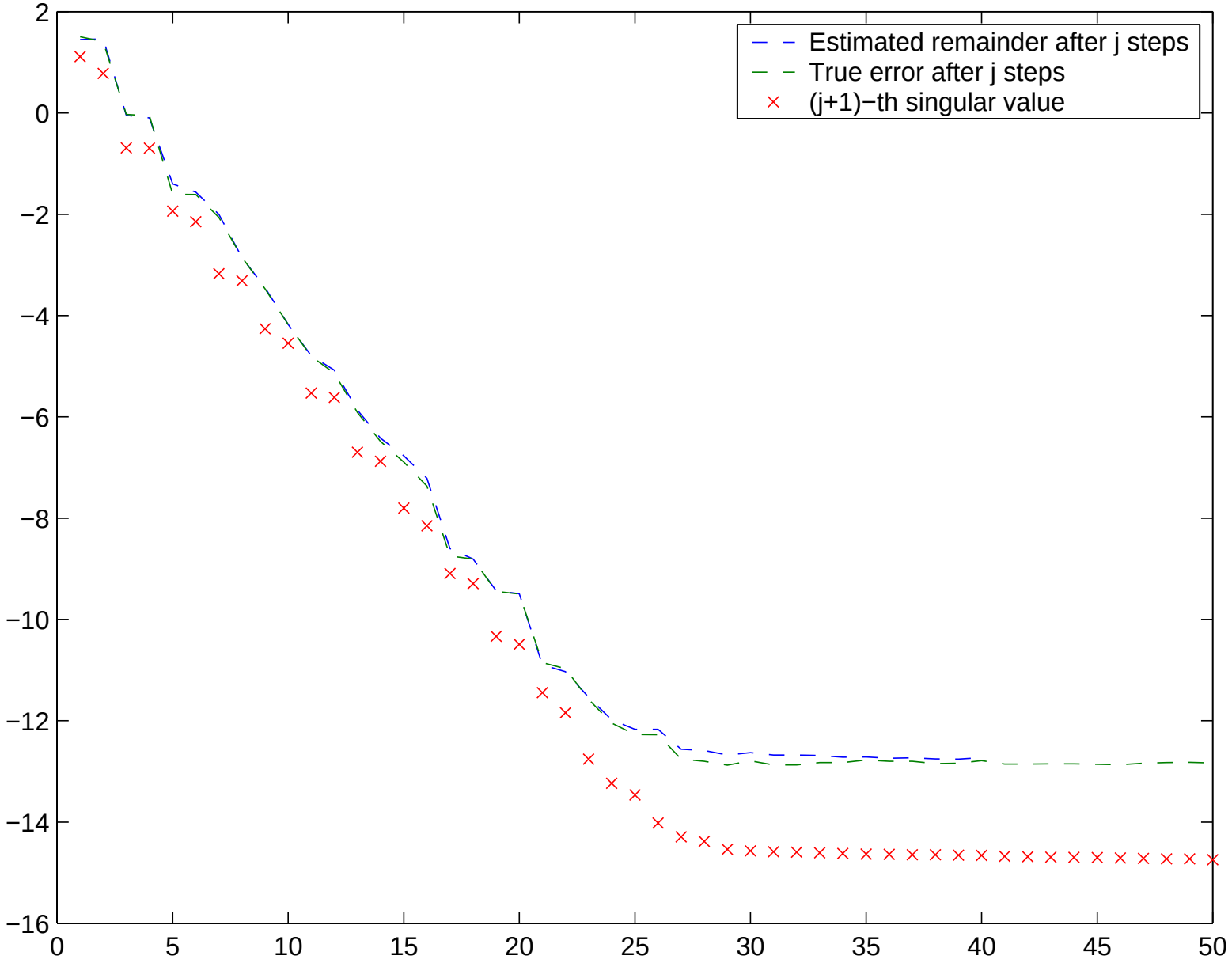
$$e_l = ||(I - Q_l\,Q_l^{\mathrm{t}})\,A||.$$

Notice that in reality, we can rarely afford to compute e_l.
Instead, we compute something like

$$f_l = \left|\left|(I - Q_l\,Q_l^{\mathrm{t}})\,[y_{l+1}, y_{l+2}, \cdots, y_{l+10}]\frac{1}{10}\right|\right|_{\text{Frobenius}}.$$

Our estimate for the rank is the lowest integer l such that $f_l < \varepsilon$.

(Notice that plenty of operations here can be optimized. A lot.)



$\varepsilon = 10^{-10}$ True ε-rank $= 19$ Estimated ε-rank $= 21$ / 19.

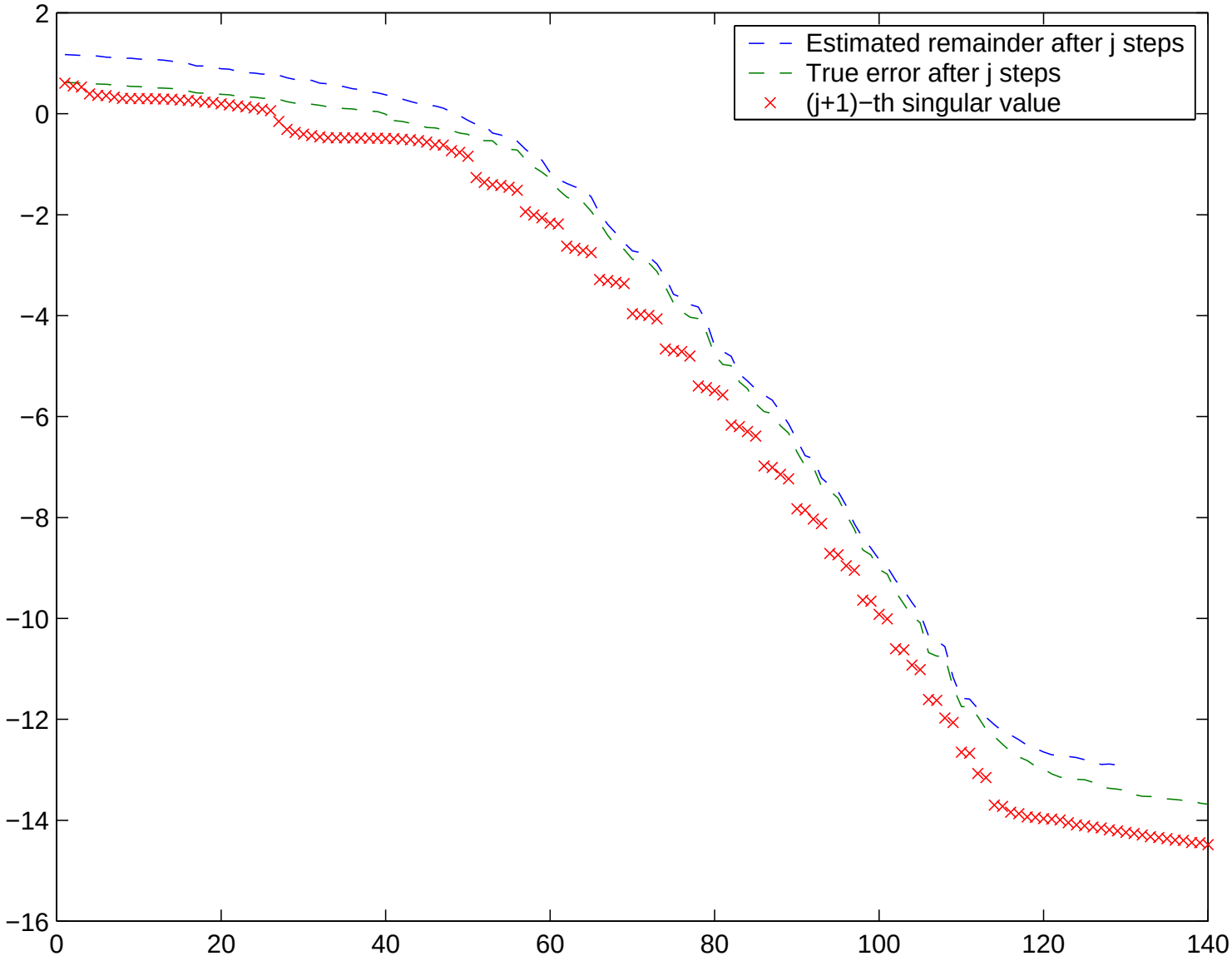
Was this just a lucky realization?

We ran the algorithm a million times and got these estimated ranks:

```
k = 17:          0 times
k = 18:          0 times
k = 19:       4178 times
k = 20:     246905 times
k = 21:     664486 times
k = 22:      81789 times
k = 23:       2634 times
k = 24:          8 times
k = 25:          0 times
k = 26:          0 times
```

The numbers above relate to the *initial estimate* of the rank. The second estimate was *always* 19, and the error was *always* less than 10^{-10}.

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



$\varepsilon = 10^{-10}$ True ε-rank $= 101$ Estimated ε-rank $= 106$ / 101.

Theorem: *Let A be an $m \times n$ matrix and let k be an integer.*

Let l be an integer such that $l \geq k$.

Let Ω be an $n \times l$ matrix with i.i.d. Gaussian elements.

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

Let σ_{k+1} denote the $(k+1)$'th singular value of A.

Then

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{l\,m}\,\sigma_{k+1},$$

with probability at least

$$1 - f(l-k),$$

where f is a decreasing function satisfying

$$f(8) < 10^{-5}$$
$$f(20) < 10^{-17}.$$

Mark described how to prove this theorem. More details can be found in our paper.

An informal argument for "why" it works springs from the SVD. Recall

$$A = U\, D\, V^{\mathrm{t}}.$$

We have

$$y = A\,\omega = U\, D\, V^{\mathrm{t}}\,\omega.$$

Set $\nu = V^{\mathrm{t}}\,\omega$. Then

$$y = U\, D\,\nu = \nu_1\,\sigma_1\,u_1 + \nu_2\,\sigma_2\,u_2 + \cdots \nu_n\,\sigma_n\,u_n.$$

We see that y is a linear combination of the left singular vectors of A, *weighted according to the corresponding singular values*.

Recall the error bound:

$$||A - Q\,Q^{\mathrm{t}}\,A||_2 \leq 10\,\sqrt{l\,m}\,\sigma_{k+1},$$

The high-lighted factor is somewhat undesirable for a couple of reasons:

- The algorithm cannot determine the ε-rank if ε is too close to the computational precision.

- There could be problems in cases where the singular values decay slowly.

Important: In the applications that we have in mind, the singular values decay exponentially. In such cases, the only effect of the $\sqrt{lm}$ factor is that a couple too many random vectors may be generated. *The computed decomposition is still accurate to precision ε.*

How does Algorithm I perform when we do not have a fast method for applying A to a vector?

When $k \ll \min(m, n)$, Algorithm 1 might be slightly faster than Gram-Schmidt:

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

Other potential benefits:

- Data-movement.
- Parallelization.

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

Algorithm 2: An $O(m\,n\,log(k))$ algorithm for *general* matrices:

Work by Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert.
(The speaker was — much to his regret — not involved with this development.)

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

$$\underset{m \times l}{Y} = \underset{m \times n}{A}\ \underset{n \times l}{\Omega}.$$

Key points:

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

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 $n\ \log(l)$ operations.

What is this "random but structured" matrix Ω?

$$\begin{array}{ccccc} \Omega & = & D & F & S \\ n \times l & & n \times n & n \times n & n \times l \end{array}$$

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) and others.

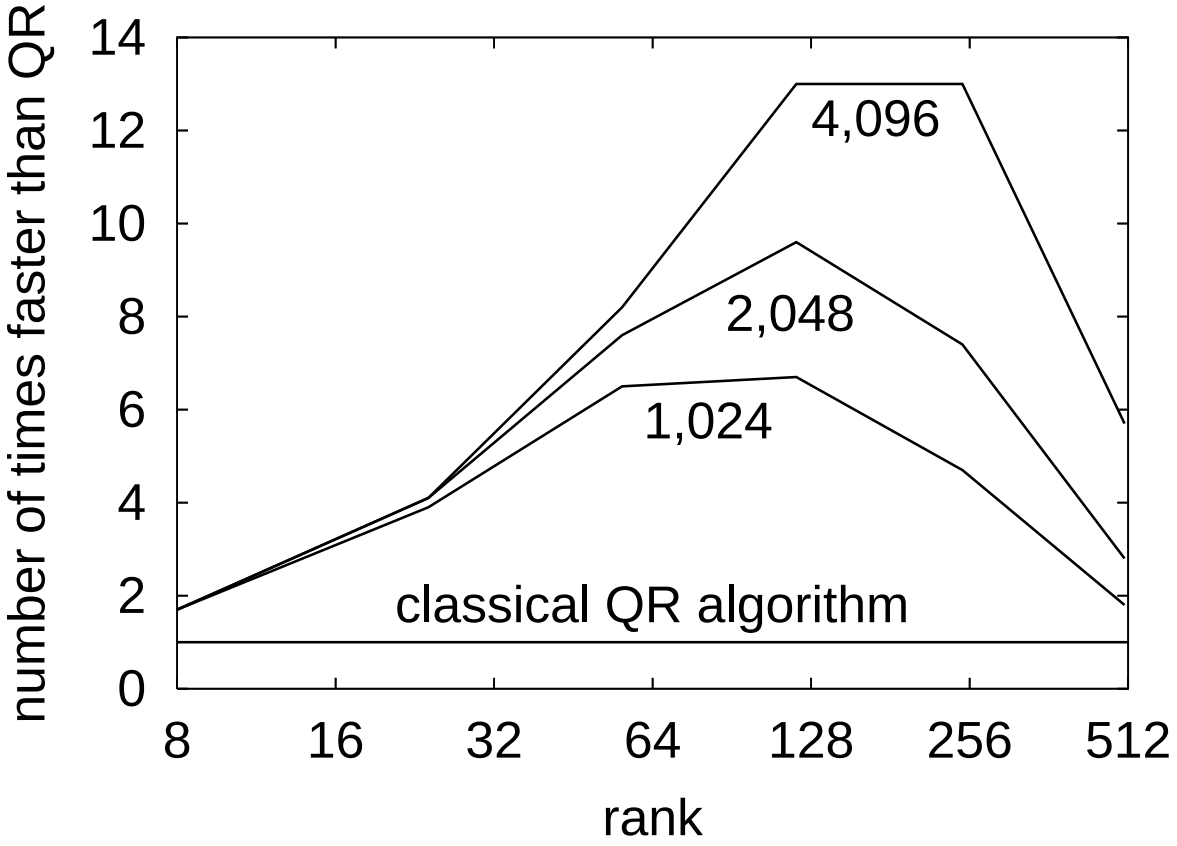
What is the probability of failure?

The proofs obtained so far do not assure quite as high likelihood of success as the proofs for Algorithm 1 did. (Say $1 - 10^{-7}$ instead of $1 - 10^{-17}$.)

The proofs may not be sharp however. An indication that this may be the case is that the algorithm has never failed during testing.

Should it prove to be the case that Algorithm 2 occasionally fails, a cheap verification can be put in place. (Simply note that the difference between A and the computed approximation to A can rapidly be applied to a vector.)

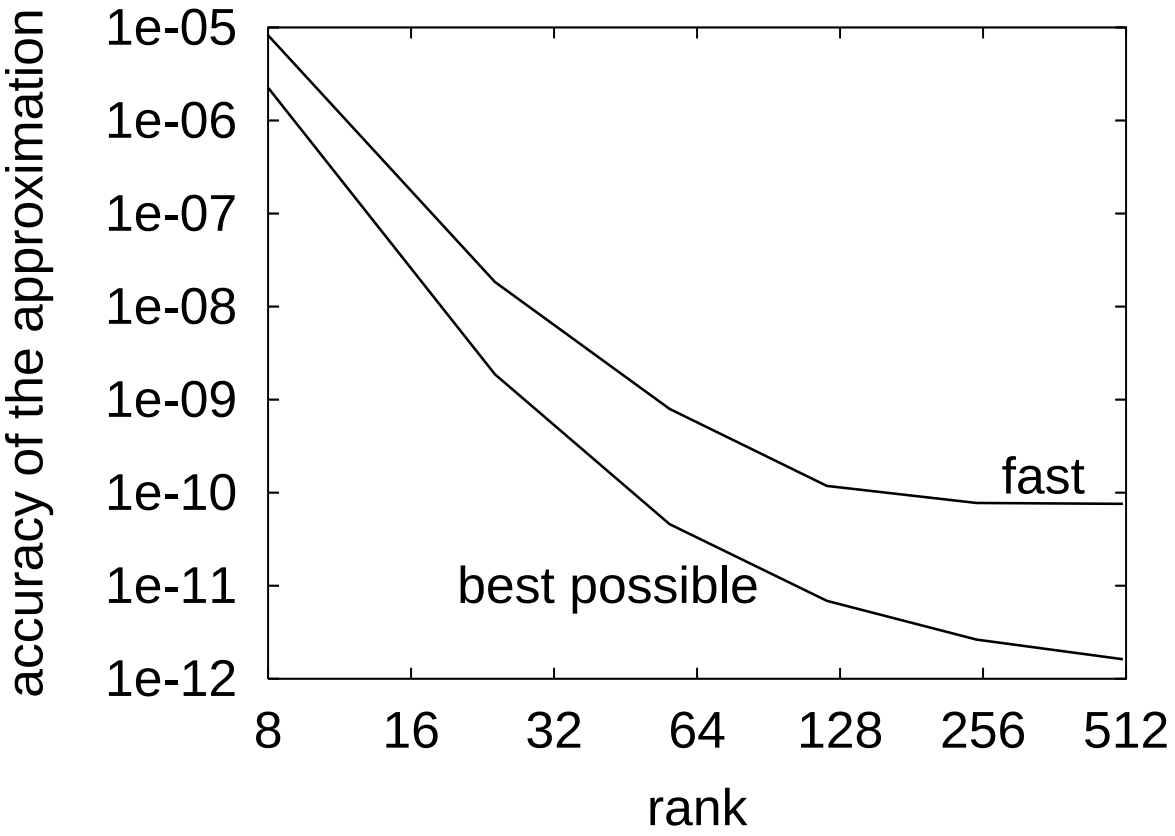
Speed gain on square matrices of various sizes



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

This slide comes from a talk by Mark Tygert.

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.

This slide comes from a talk by Mark Tygert.

Key points — Algorithm 2:

(Franco Woolfe, Edo Liberty, Vladimir Rokhlin, Mark Tygert)

There exists an algorithm for rank-k matrix approximation (or for computing the top k singular values and vectors) with advantages over the classical pivoted $Q\,R$ algorithms such as Gram-Schmidt:

1. Substantially faster (for most ranks k of the approximation), costing $O(n^2 \ln(k) + nk^2)$ — not $O(n^2 k)$ — for an $n \times n$ matrix.

2. Uses less storage when the input matrix is to be preserved, especially for matrices evaluated on-the-fly.

3. Operates reliably and accurately on any matrix.

4. Parallelizes naturally.

A technical point:

A relatively unknown cousin of the SVD or the QR decompositions is the "interpolatory decomposition". It is in this environment frequently cheaper and more convenient.

An interpolatory decomposition of an $m \times n$ matrix A of rank k, is a factorization

$$\underbrace{A}_{m\times n} = \underbrace{S}_{m\times k}\ \underbrace{A_{\text{row}}}_{k\times n}$$

where

- A_{row} consists of k rows of A,
- S is a matrix that contains a $k \times k$ unity matrix, and,
- no entry of S is larger than 2.

Notes on interpolatory decompositions:

- Every matrix has one.
- In typical situations, the decomposition is highly non-unique.
- An algorithm for constructing one that typically takes $O(m\,n\,k)$ (and assuredly never more than $O(m\,n^2)$) is given by Gu and Eisenstat.
- In practise, the Gu/Eisenstat algorithm is unnecessarily complicated. Carefully pivoted Gram-Schmidt on the rows works fine and costs $\sim 2\,m\,n\,k$. (Matrices for which Gram-Schmidt fails *can* be constructed.)
- Some authors cause themselves head-aches by insisting on finding the absolutely optimal k. In many applications, this does not matter much since the cost of very slightly over-estimating the rank is low. (Generally speaking, an application is "safe" if the singular values decay appropriately fast.)

Using interpolatory approximations, it is simple to obtain a matrix factorization of a matrix A from a basis for the column space of A. To illustrate, suppose that we have found a matrix B such that

$$A \approx B\,C,$$

where C is a matrix that we do not know.

Then at a cost of $O(m\,k^2)$ we determine k rows of B that form a well-conditioned basis for the row-space of B. Collecting these into the matrix B_{row}, we obtain

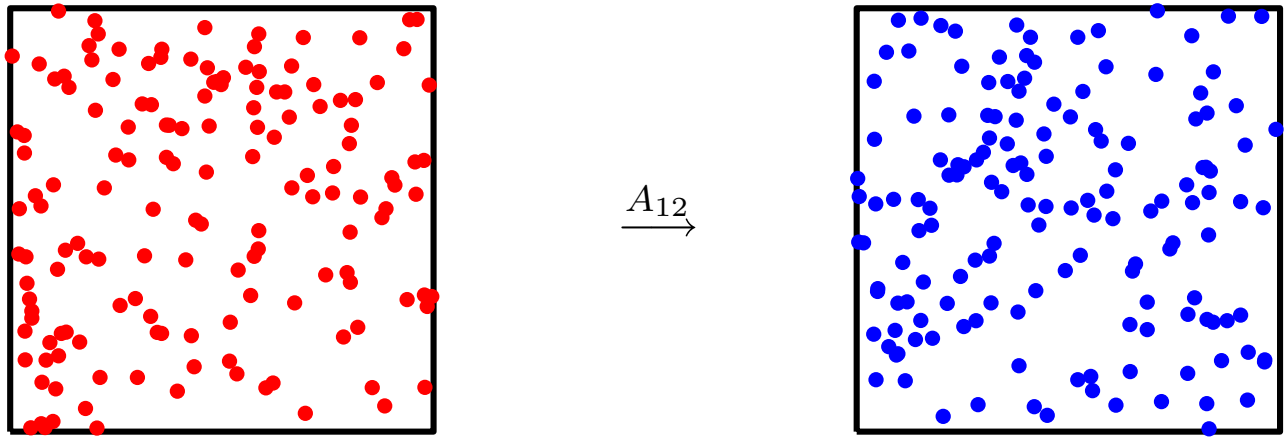
$$B = P \begin{bmatrix} I_k \\ T \end{bmatrix} B_{\text{row}},$$

where P is a permutation matrix. Then

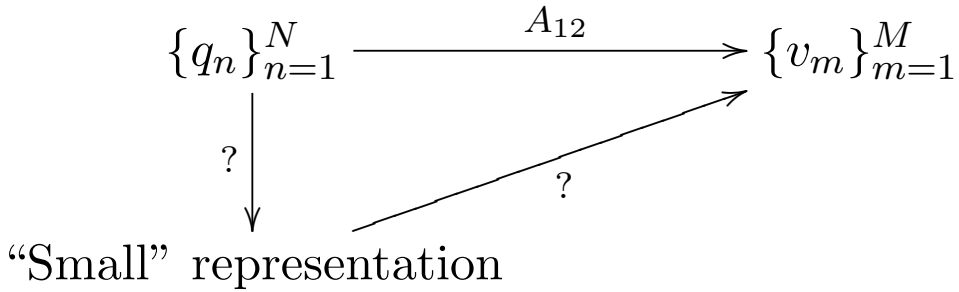
$$A \approx B\,C = P \begin{bmatrix} I_k \\ T \end{bmatrix} B_{\text{row}}\,C = P \begin{bmatrix} I_k \\ T \end{bmatrix} A_{\text{row}},$$

where A_{row} consists of k rows of A.

So, once B is determined, only an $O(m\,k^2)$, or possibly $O((m+n)\,k^2)$, cost remains.

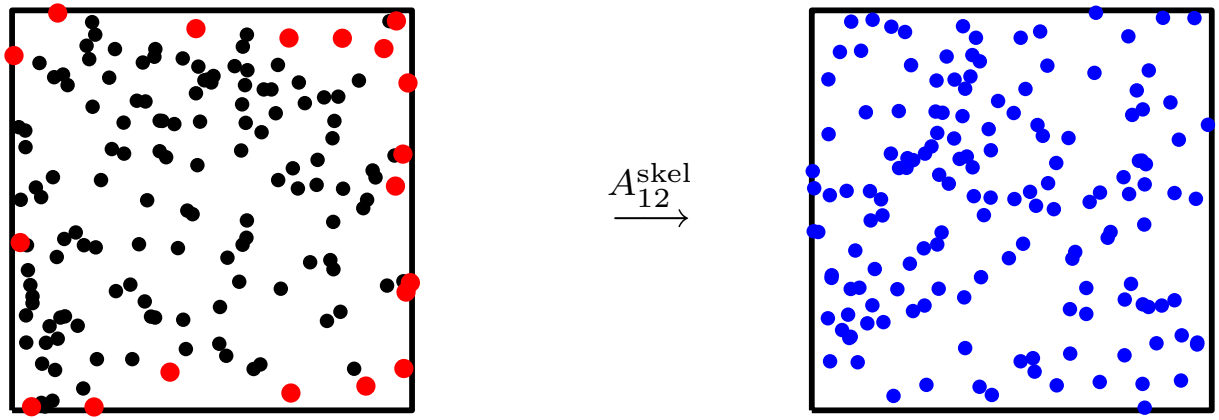


Sources $\{q_n\}_{n=1}^N$ Potentials $\{v_m\}_{m=1}^M$



The key observation is that $k = \text{rank}(A_{12}) < \min(M, N)$.

Skeletonization

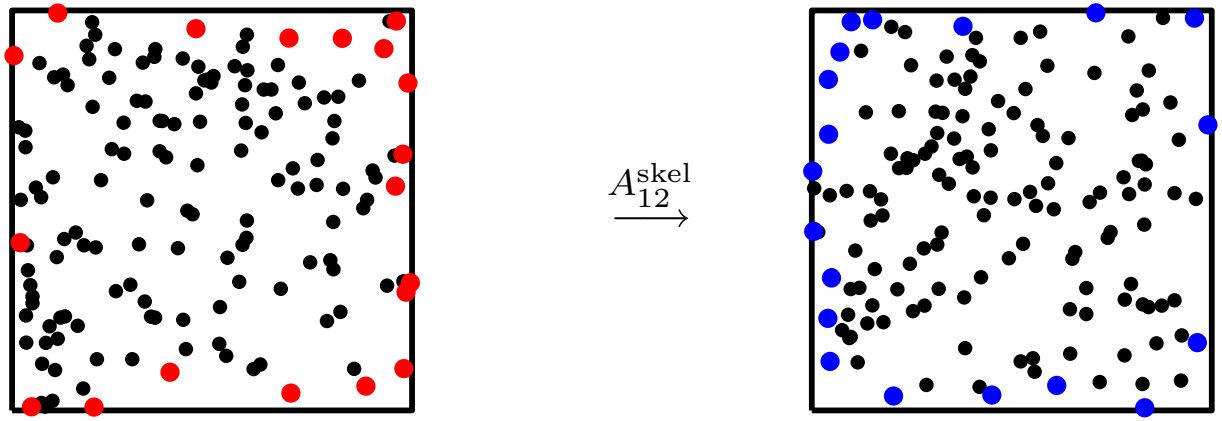


$$\begin{array}{ccc} \{q_n\}_{n=1}^{N} & \xrightarrow{\quad A_{12} \quad} & \{v_m\}_{m=1}^{M} \\ \Big\downarrow U_2^{\mathrm{t}} & \nearrow_{A_{12}^{\text{skel}}} & \\ \{\tilde{q}_{n_j}\}_{j=1}^{k} & & \end{array}$$

We can pick k points in Ω_{S} with the property that any potential in Ω_{T} can be replicated by placing charges on these k points.

- The choice of points does not depend on $\{q_n\}_{n=1}^{N}$.
- A_{12}^{skel} is a submatrix of A_{12}.

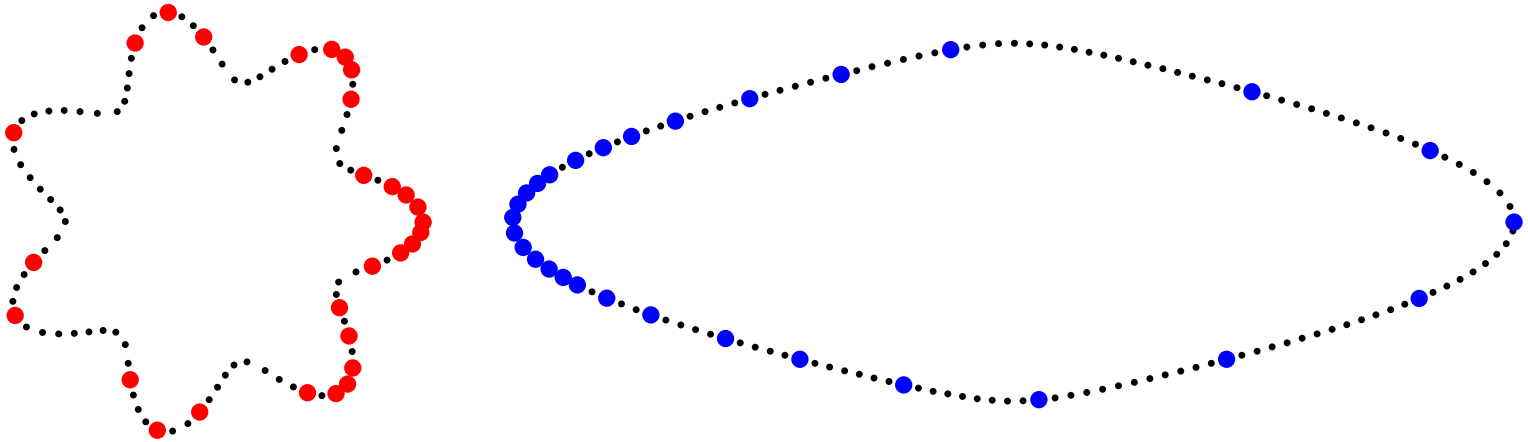
We can "skeletonize" both Ω_1 and Ω_2.



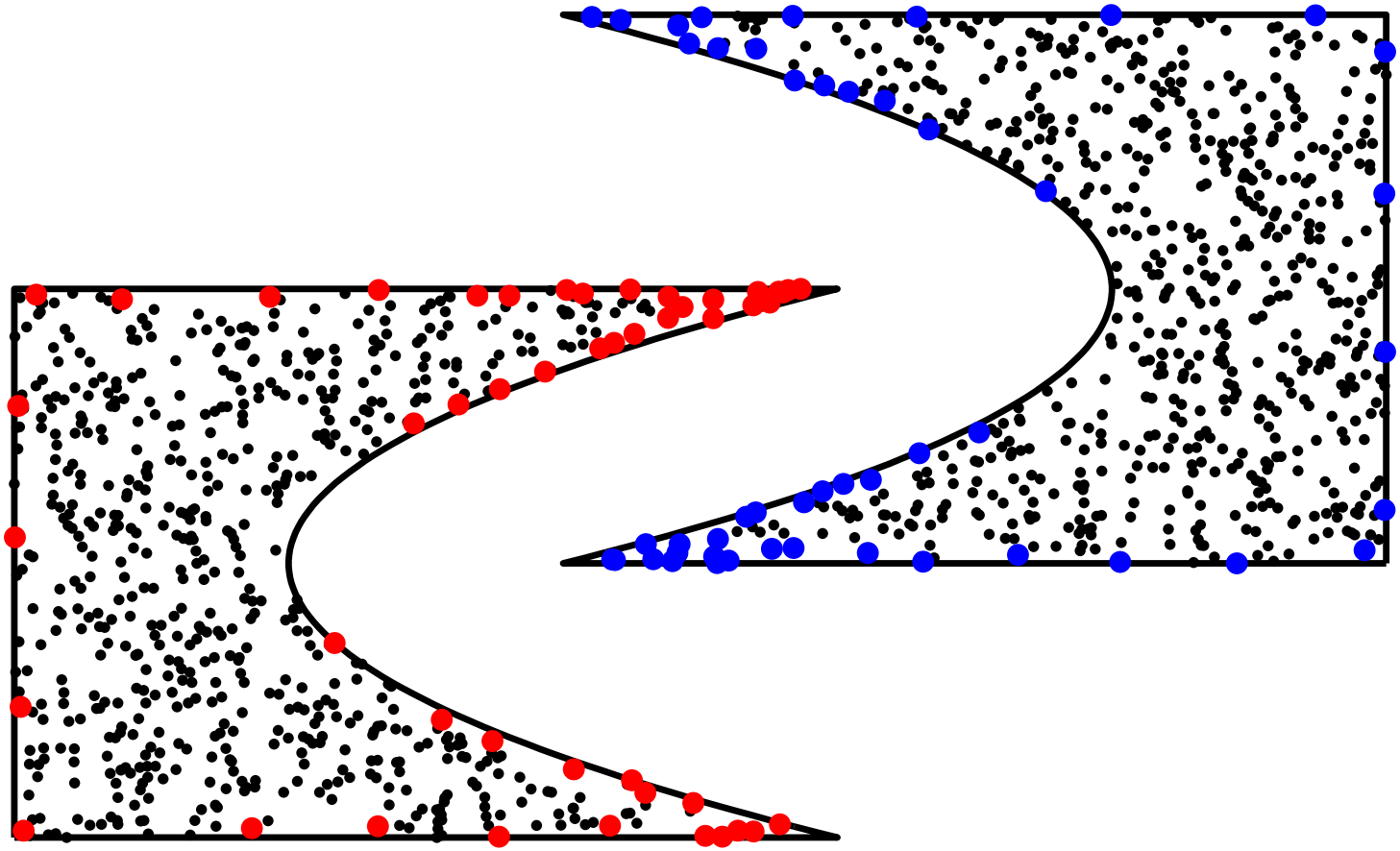
$$\begin{array}{ccc} \{q_n\}_{n=1}^{N} & \xrightarrow{\quad A_{12} \quad} & \{v_m\}_{m=1}^{M} \\ \Big\downarrow U_2^{\mathrm{t}} & & \Big\uparrow U_1 \\ \{\tilde{q}_{n_j}\}_{j=1}^{k} & \xrightarrow[\quad A_{12}^{\text{skel}} \quad]{} & \{v_{m_j}\}_{j=1}^{k} \end{array}$$

Rank $= 19$ at $\varepsilon = 10^{-10}$.

Skeletonization can be performed for Ω_{S} and Ω_{T} of various shapes.

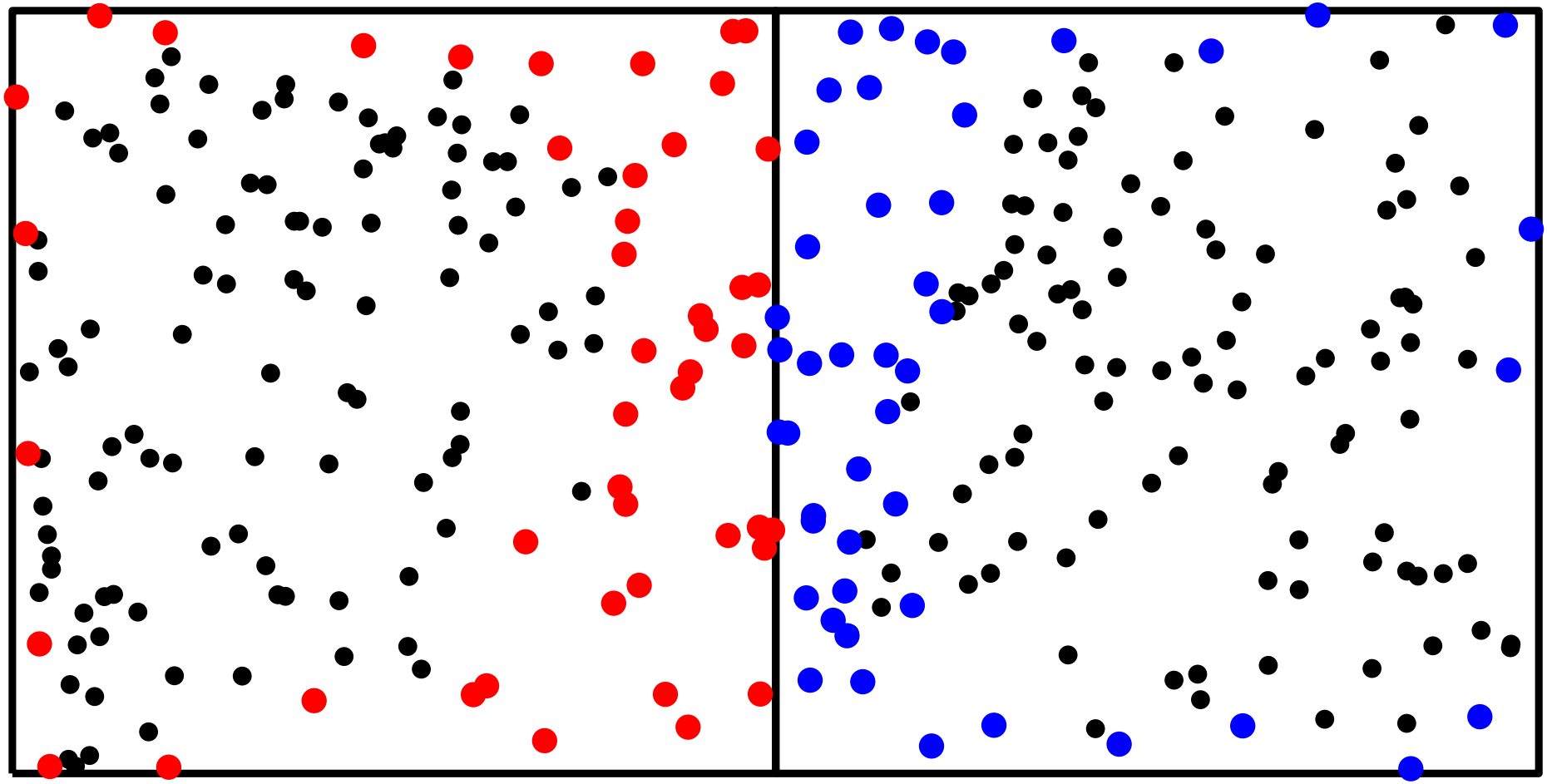


Rank $= 29$ at $\varepsilon = 10^{-10}$.

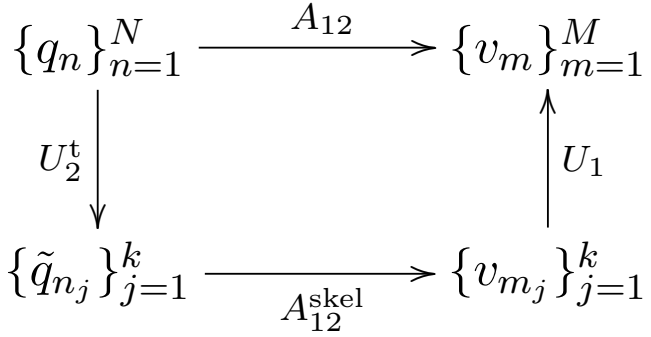


Rank $= 48$ at $\varepsilon = 10^{-10}$.

Adjacent boxes can be skeletonized.



Rank $= 46$ at $\varepsilon = 10^{-10}$.



Benefits:

- The rank is optimal.
- The projection and interpolation are cheap.
  U_1 and U_2 contain $k \times k$ identity matrices.
- The projection and interpolation are well-conditioned.
- Finding the k points is cheap.
- The map $\tilde{A}_{12}$ is simply a restriction of the original map A_{12}.
  (We can loosely say that "the physics of the problem is preserved".)
- Interaction between **adjacent** boxes can be compressed
  (no buffering is required).

Work in progress:

- Develop efficient strategies for determining the rank adaptively, and for updating the ON-basis for the column space.
- Simplify proofs.
- Investigate how different choices of random sampling vectors influence the performance. Random vs. pseudo-random.

Applications:

- Fast algorithms for matrix algebra (matrix-vector multiplies, matrix-inversions, spectral decompositions) involving differential and integral operators.
- Eigen-decompositions of discrete Laplacians.
- Multiscale modeling.
- Analysis of network matrices (data mining).