Randomized methods for the approximation of matrices

Gunnar Martinsson, The University of Colorado at Boulder

Joint work with Vladimir Rokhlin and Mark Tygert.

Definition: We say that an $m \times n$ matrix B has rank k if there exist an $m \times k$ matrix C and a $k \times n$ matrix D such that

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

In other words, the column and the row spaces of B have dimension (at most) k.

Definition: Given a vector $v \in \mathbb{R}^n$, we define its norm by

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

Definition: Given a matrix A, we define its norm by

$$||A|| = \sup_{||x||=1} ||A\,x||.$$

Approximation by low rank matrices – problem formulation:

Let A be an $m \times n$ matrix.

Let ε be a small real number denoting computational accuracy. Say $\varepsilon = 10^{-10}$.

We seek to construct an orthonormal basis for the column space of A accurate to within precision ε. In other words, we seek to construct a matrix $V = [v_1, v_2, \ldots, v_k]$ with orthonormal columns such that

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

We seek a basis with a small, but not necessarily absolutely optimal, k.

Note: Once you have a basis for the column space of a matrix, any "standard" factorization (LU, QR, SVD, *etc*) can be computed cheaply.

Note: We have in mind here matrices that are large, but for which (1) holds for relatively small k. Such matrices are important in applications.

Why are low-rank approximations interesting?

Accelerating numerics:

The factorization can be used to accelerate linear algebraic operations such as matrix-vector multiplies, matrix-matrix multiplies, etc. Tomorrow's talk!

They reveal properties of the matrix:

For instance, spectral properties associated with the dominant eigenvalues or singular values can be determined.

Statistical analysis:

Suppose that each column of A is a sample of m random variables from a multivariate normal distribution. Then the dependencies among these variables can be determined via "principal component analysis".

Analysis of network matrices:

Suppose that A is a network matrix representing for instance the link structure of the World Wide Web. Then approximate low rank matrix approximations can be used to analyze the network. Example: The Google "page rank" algorithm.

There is a "standard" technique for constructing an ON-basis for the column space: Gram-Schmidt.

Recall our $m \times n$ matrix $A = [a_1, a_2, \ldots, a_n]$.

Step 1: $v_1 = a_1/||a_1||$

Step 2: $\hat{v}_2 = a_2 - (a_2 \cdot v_1)\, v_1$

$v_2 = \hat{v}_2/||\hat{v}_2||$

Step 3: $\hat{v}_3 = a_3 - (a_3 \cdot v_1)\, v_1 - (a_3 \cdot v_2)\, v_2$

$v_3 = \hat{v}_3/||\hat{v}_3||$

etc

For stability reasons, one must in real applications use a variation of the above algorithm called "modified Gram-Schmidt".

The Gram-Schmidt process is just one of many well-established techniques for computing a basis for the column space of a given matrix. Some of these go by names such as "Householder", "Givens", *etc.*

The computational cost of all such traditional methods satisfies

$$\text{COST} \sim c\,m\,n\,k,$$

where, recall, m and n denote the size of the matrix, k is its approximate rank, and c is a small constant. We say that these methods are

$$O(m\,n\,k).$$

Goal: Techniques whose asymptotic cost is lower than $O(m\,n\,k)$.

We will consider two different environments.

First we consider the case where we have a fast algorithm for computing matrix-vector products. In other words, the cost T_{mult} for evaluating

$$x \mapsto A\,x$$

is smaller than $O(m\,n)$. Typically,

$$T_{\text{mult}} = O(m+n)$$

or

$$T_{\text{mult}} = O((m+n)\,\log(m+n)).$$

(We will return to general matrices later ...)

Examples of when $x \mapsto A\,x$ can be evaluated rapidly:

- A is sparse.
  For instance, A may be a discrete approximation to a difference operator.
  Or, A may be a network matrix (describing *e.g.* the link structure of the WWW).

- A is sparse in Fourier space.
  Then the FFT can be used to rapidly compute $x \mapsto A\,x$.

- A is a discrete approximation to an integral operator of classical potential theory. Then there frequently exist fast methods for evaluating the map $x \mapsto A\,x$. One example is the "Fast Multipole Method".

- $A = B^{-1}$ for some sparse matrix B.
  Then computing $y = A\,x = B^{-1}\,x$ amounts to solving the system $B\,y = x$.
  Frequently, there exist fast algorithms for this task.

There is a well-established method for this problem: Lanczos.

This algorithm is sometimes $O(T_{\text{mult}}\, k)$, but the method that we will describe is more robust, faster, and more easily parallelizable.

The method that we propose is based on randomized sampling.

This means that it has a non-zero probability of giving an incorrect answer.

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^{-10} are standard.

Definition: We say that a vector $\tilde{\omega} \in \mathbb{R}^n$ is a random direction vector if

$$\tilde{\omega} = \frac{1}{\left(\sum_{j=1}^{n} |\omega_j|^2\right)^{1/2}} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix},$$

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

Alternatively, random direction vectors can be defined as vectors drawn from a uniform random distribution on the surface of the unit sphere in $\mathbb{R}^n$.

Note: In practise, the vectors $\tilde{\omega}$ will be constructed using "random number generators". The quality of the generators will not matter much.
(Astonishingly little, in fact.)

Algorithm 1: *Rapid computation of a low-rank appoximation.*	• Let ε denote the computational accuracy desired. • Let A be an $m \times n$ matrix of ε-rank k. • We seek a rank-k approximation of A. • We can perform matrix-vector multiplies fast.

Let $\omega_1, \omega_2, \ldots$ be a sequence of vectors in $\mathbb{R}^n$ whose entries are i.i.d. random variables drawn from a standardized Gaussian distribution.

Form 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.

Algorithm 1: *Rapid computation of a low-rank appoximation.*	• Let ε denote the computational accuracy desired. • Let A be an $m \times n$ matrix of ε-rank k. • We seek a rank-k approximation of A. • We can perform matrix-vector multiplies fast.

Let $\omega_1, \omega_2, \ldots$ be a sequence of vectors in $\mathbb{R}^n$ whose entries are i.i.d. random variables drawn from a standardized Gaussian distribution.

Form 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.

How to measure "how well we are doing":

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

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

Orthonormalize the vectors $[y_1, y_2, \ldots, y_l]$ and collect the result in the matrix V_l.

The "error" after l steps is then

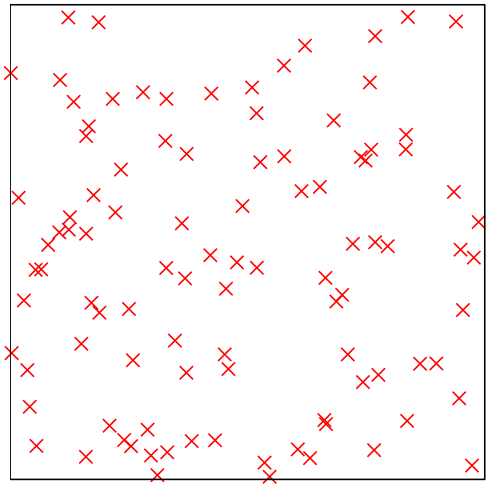
$$e_l = ||(I - V_l\, V_l^{\mathrm{t}})\, A||.$$

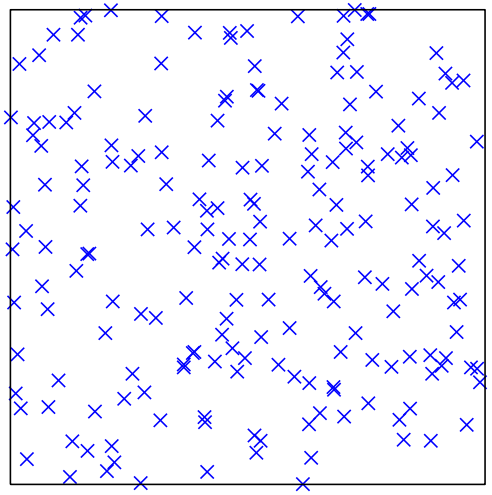
The quantity e_l should be compared to the minimal error

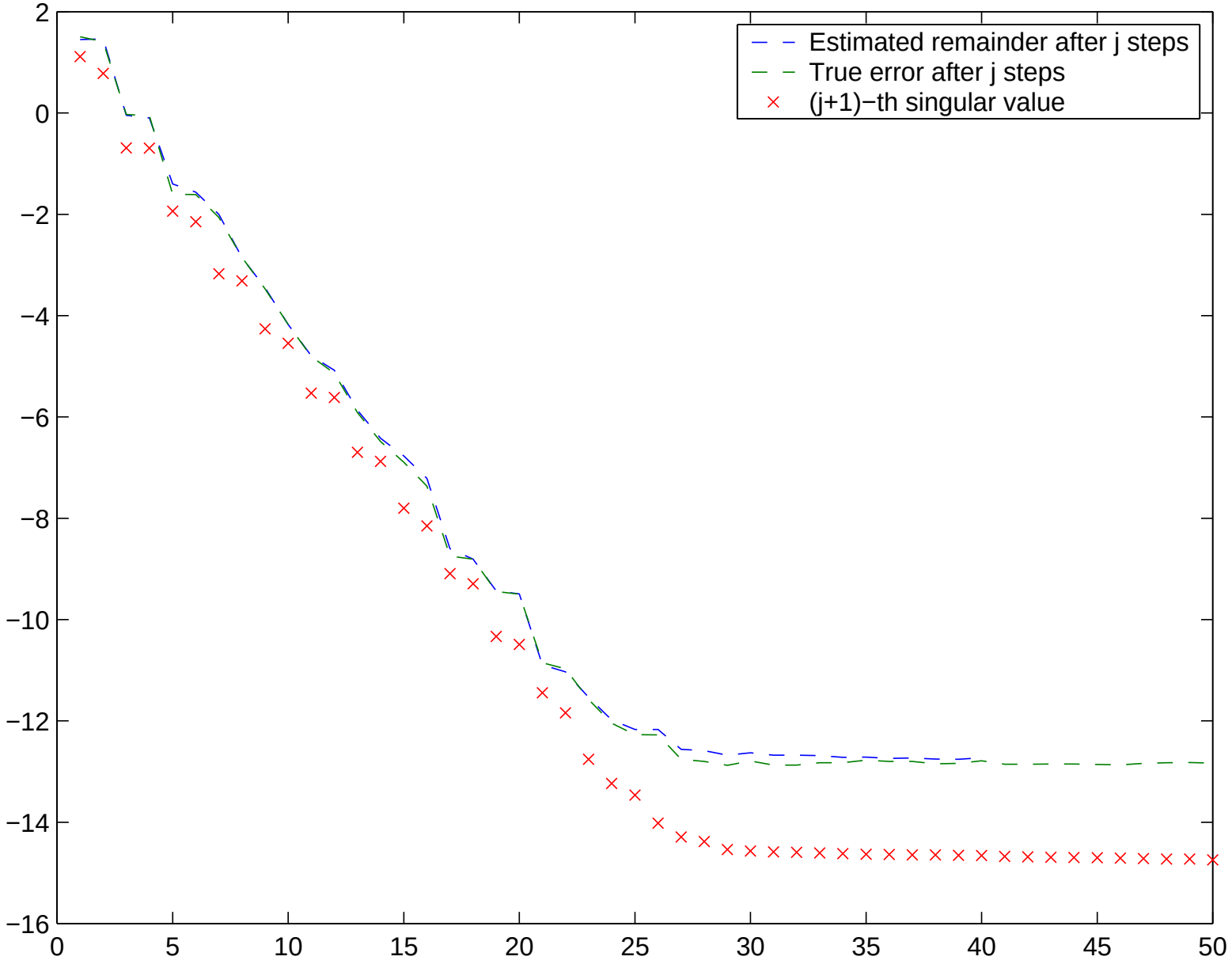
$$\sigma_{l+1} = \min_{\mathrm{rank}(B)=l} ||A - B||.$$

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

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 $\mathbb{R}^2$.







$\varepsilon = 10^{-10}$, exact ε-rank $= 34$, nr. of matrix-vector multiplies required $= 36$.

There is one problem . . .

There is one problem ...

The quantity

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

is very expensive to compute.

Let us revisit the algorithm ...

How to measure "how well we are doing":

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

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

Orthogonalize the vectors $[y_1, y_2, \ldots, y_l]$ and collect the result in the matrix V_l.

The "error" after l steps is then

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

How to measure "how well we are doing":

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

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

Orthogonalize the vectors $[y_1, y_2, \ldots, y_l]$ and collect the result in the matrix V_l.

The "error" after l steps is then

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

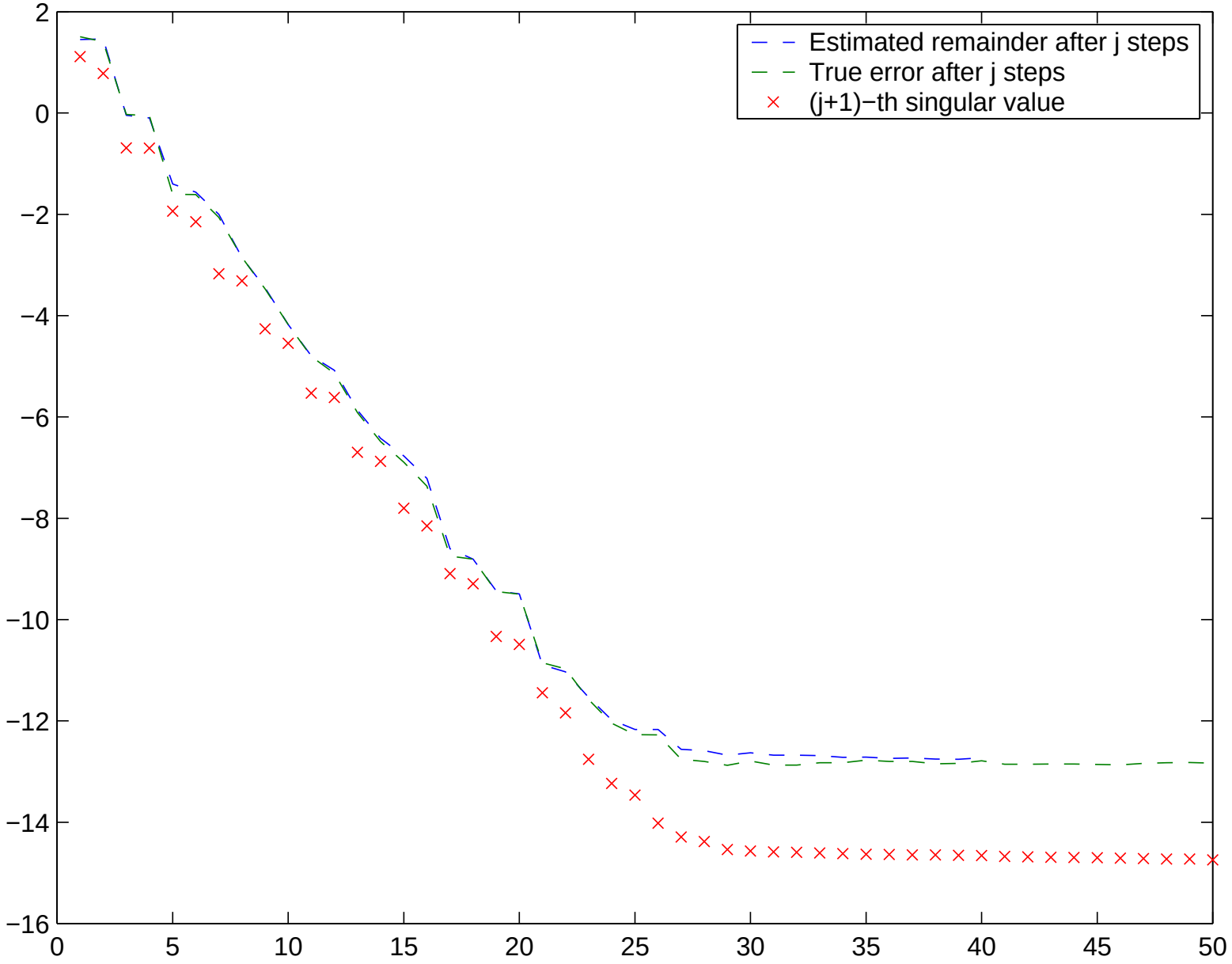
In reality, computing e_l is not affordable. Instead, we compute something like

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

The computation stops when we come to an l such that $f_l < \varepsilon/\sqrt{m}$.

The quantity e_l (estimated by f_l) should be compared to the minimal error

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



$\varepsilon = 10^{-10}$, exact ε-rank $= 34$, nr. of matrix-vector multiplies required $= 36$.

Was this just a lucky realization?

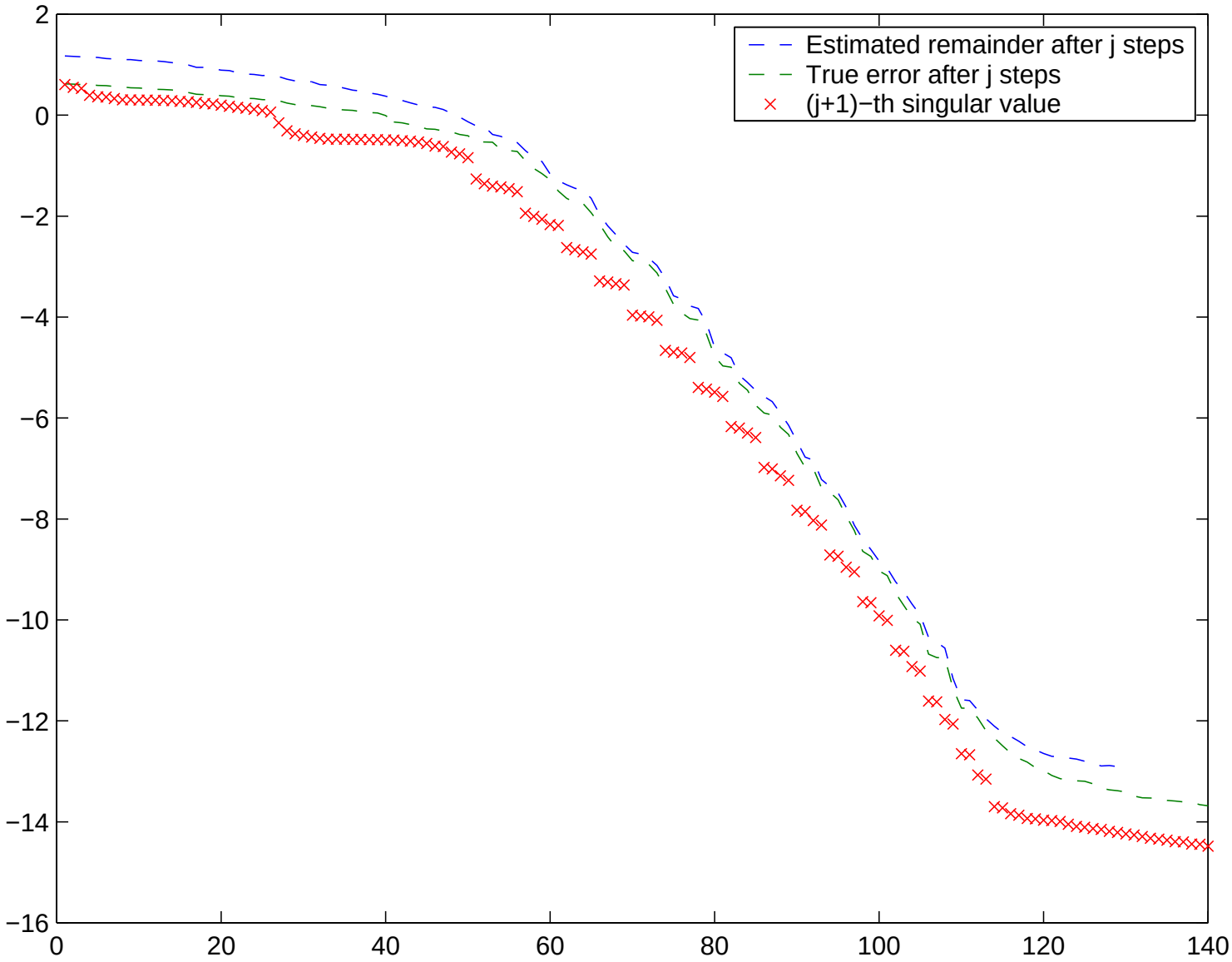
We collected statistics from 1 000 000 realizations:

(Recall that 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 $= 106$.

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

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

Then

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

with probability at least

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

where φ is a decreasing function satisfying

$$\varphi(8) < 10^{-5}$$

$$\varphi(20) < 10^{-17}.$$

We will sketch the proof for a related but simpler problem:

Consider the following situation:

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

Using randomized sampling to solve this problem, we would generate a set

$$\{\tilde{\omega}^{(1)},\ \tilde{\omega}^{(2)},\ \ldots\ ,\ \tilde{\omega}^{(q)}\},$$

of random direction vectors, and then determine a bound for $||A||$ from the set

$$\{A\,\tilde{\omega}^{(1)},\ A\,\tilde{\omega}^{(2)},\ \ldots\ ,\ A\,\tilde{\omega}^{(q)}\}.$$

Claim: Let A be an $m \times n$ matrix, let q be a positive integer, and let $\{\tilde{\omega}^{(j)}\}_{j=1}^{q}$ be random direction vectors. Then

$$10\,\sqrt{n}\,\max(||A\,\tilde{\omega}^{(1)}||,\ \ldots\ ,\ ||A\,\tilde{\omega}^{(q)}||)$$

is an upper bound for $||A||$ with probability at least $1 - 10^{-q}$.

Sketch of proof:

Suppose first that A has rank 1 and that we perform a single experiment, $q = 1$.

Preliminary calculation: A matrix A of rank 1 can be expressed as

$$A = ||A||\, u\, v^{\mathrm{t}},$$

where u and v are unit vectors. Then if $\tilde{\omega}$ is a random direction vector,

$$||A\,\tilde{\omega}|| = ||\,||A||\, u\, v^{\mathrm{t}}\,\tilde{\omega}|| = ||A||\,||u\,(v\cdot\tilde{\omega})|| = ||A||\,|v\cdot\tilde{\omega}|.$$

Let μ be real positive number. We seek to argue that if μ is small enough, then

$$\frac{1}{\mu}\,||A\,\tilde{\omega}||$$

will with high probability be an upper bound for $||A||$. The failure probability is

$$\mathbf{P}\left(\frac{1}{\mu}\,||A\,\tilde{\omega}|| \le ||A||\right) = \mathbf{P}\left(\frac{1}{\mu}\,||A||\,|v\cdot\tilde{\omega}| \le ||A||\right) = \mathbf{P}\left(|v\cdot\tilde{\omega}| \le \mu\right).$$

We need to estimate the probability that the cosine of the angle between a fixed unit vector v, and a unit vector or random orientation is less than μ.

We need to estimate the probability that the cosine of the angle between a fixed unit vector v, and a unit vector or random orientation is less than μ.

If v, $\tilde{\omega} \in \mathbb{R}^2$, then this is easy. Since the probability distribution of $\tilde{\omega}$ is isotropic, we can pick coordinates so that $v = [0, 1]^{\mathrm{t}}$. Then

$$\mathbf{P}\left(|v \cdot \tilde{\omega}| \leq \mu\right) = \frac{4 \arcsin(\mu)}{2\,\pi - 4 \arcsin \mu}.$$

Using multivariate calculus, analogous formulas can be computed in any dimension.

We will not pursue that line of proof, instead, we will estimate the probability using elementary probability theory.

We need to estimate the probability that the cosine of the angle between a fixed unit vector v, and a unit vector or random orientation is less than μ.

Due to isotropy, we can choose coordinates so that $v = [0, 0, \ldots, 0, 1]^{\mathrm{t}}$. Then

$$\mathbf{P}\left(|v \cdot \tilde{\omega}| \leq \mu\right) = \mathbf{P}\left(\frac{|\omega_n|}{\left(\sum_{j=1}^{n} |\omega_j|^2\right)^{1/2}} \leq \mu\right),$$

where the ω_j are i.i.d. normalized Gaussian random variables.

Now use that $\sum_{j=1}^{n} |\omega_j|^2$ has a χ^2 distribution with expectation n *and standard deviation* $\sqrt{2n}$. Consequently,

$$\mathbf{P}\left(|v \cdot \tilde{\omega}| \leq \mu\right) \approx \mathbf{P}\left(\frac{|\omega_n|}{\sqrt{n}} \leq \mu\right) = \int_{-\mu\sqrt{n}}^{\mu\sqrt{n}} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx \leq \frac{2\mu\sqrt{n}}{\sqrt{2\pi}} \leq \mu\sqrt{n}.$$

Recall that we have proved:

(1) The failure probability satisfies $$\mathbf{P}\left(\frac{1}{\mu}\,||A\,\tilde{\omega}|| \leq ||A||\right) = \mathbf{P}\left(|v\cdot\tilde{\omega}| \leq \mu\right).$$

(2) $$\mathbf{P}\left(|v\cdot\tilde{\omega}| \leq \mu\right) \leq \mu\sqrt{n}.$$

Combining (1) and (2), and setting $\mu = 1/(10\,\sqrt{n})$, we find that

(3) $$\mathbf{P}\left(10\sqrt{n}\,||A\,\tilde{\omega}|| \leq ||A||\right) \leq \frac{1}{10}.$$

Equation (3) concerns a single experiment. It follows that if we perform q independent experiments, then

$$10\sqrt{n}\,\max\left(||A\,\tilde{\omega}^{(1)}||,\,||A\,\tilde{\omega}^{(2)}||,\,\ldots,\,||A\,\tilde{\omega}^{(q)}||\right)$$

is an upper bound for A with probability at least

$$1-\left(\frac{1}{10}\right)^{q}.$$

This concludes the proof *for the case of rank-1 matrices*.

To complete the proof, we will prove that rank-1 matrices *is the worst case*.

To do this, we will use the following result:

Theorem: Any $m \times n$ matrix A of rank k can be written

$$A = \sum_{j=1}^{k} \sigma_j \, u^{(j)} \, (v^{(j)})^{\mathrm{t}},$$

where the vectors $\{u^{(j)}\}_{j=1}^{k}$ are ON-vectors in $\mathbb{R}^m$, the vectors $\{v^{(j)}\}_{j=1}^{k}$ are ON-vectors in $\mathbb{R}^n$, and $\{\sigma_j\}_{j=1}^{n}$ are real numbers such that

$$||A|| = \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \geq \sigma_k \geq 0.$$

It then follows from Pythagoras' theorem that

$$||A\,\tilde{\omega}|| = ||\sum_{j=1}^{k} \sigma_j \, u^{(j)} \, (v^{(j)} \cdot \tilde{\omega})|| = \left(\sum_{j=1}^{k} ||\sigma_j \, u^{(j)} \, (v^{(j)} \cdot \tilde{\omega})||^2 \right)^{1/2} \geq ||\sigma_1 \, u^{(1)} \, (v^{(1)} \cdot \tilde{\omega})||.$$

We are done!

The computational cost of Algorithm I is $O(T_{\text{mult}}\, k + m\, k^2)$.

How does Algorithm I perform when we do not have a fast method for applying A to a vector? In this case, $T_{\text{mult}} = mn$.

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

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

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 $\mathbb{C}$.

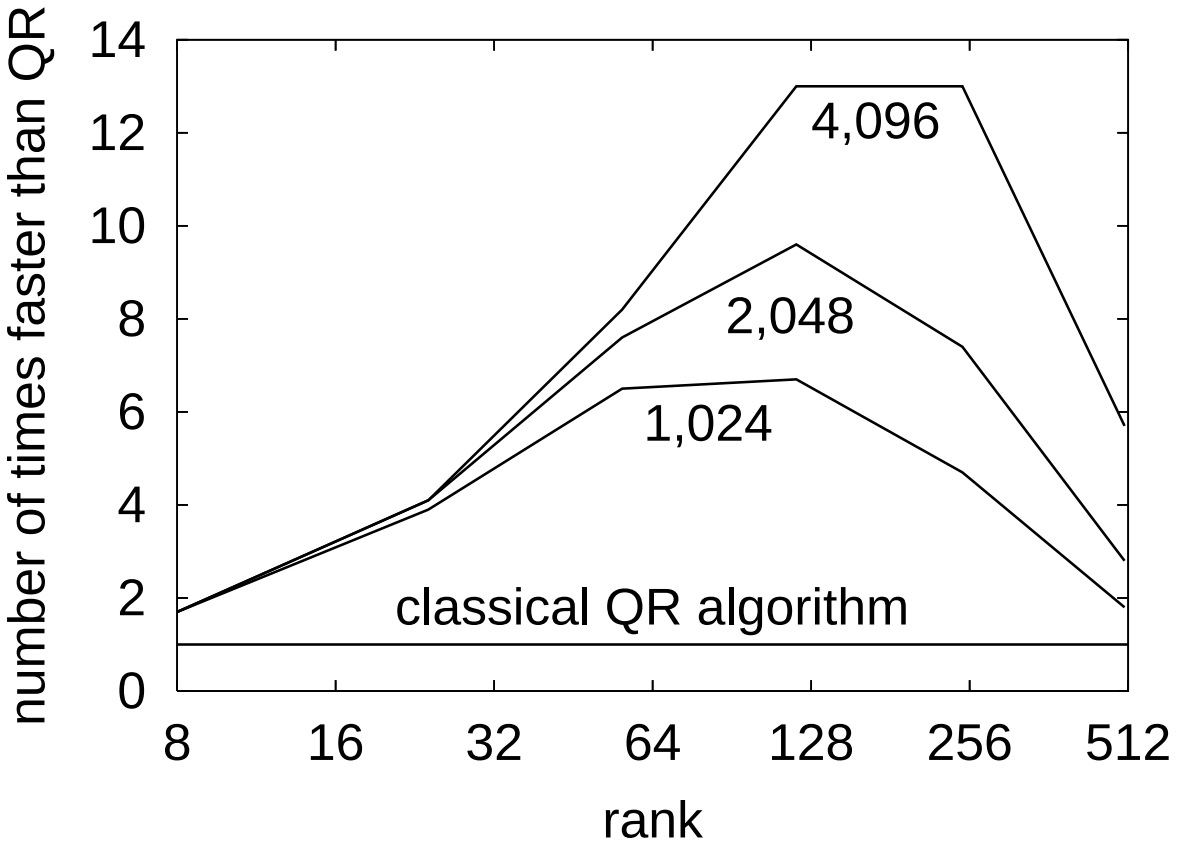
- F is the discrete Fourier transform, $F_{jk} = e^{-2\pi i(j-1)(k-1)/n}$.

- 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.

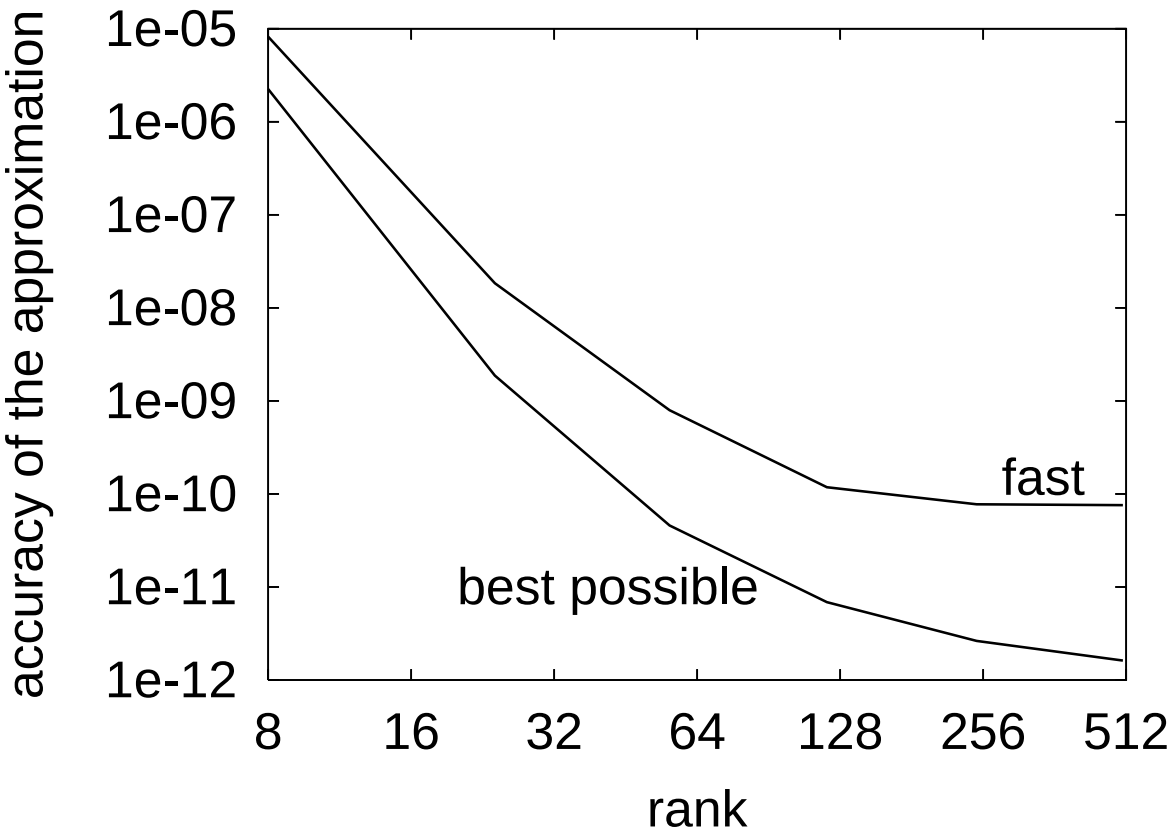
This idea was described by Nir Ailon and Bernard Chazelle (2006).
There is also related recent work by Sarlós (on randomized regression).

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.

Summary:

We solved the problem of how to construct an approximate basis for the column space of a given matrix A.

The basic idea was to study the action of A on a sequence of randomly generated vectors. This "samples" the column space of A extremely efficiently.

We studied two different environments

1. We can rapidly evaluate $x \mapsto A\,x$.

   For this environment, we constructed a method that is more robust than existing methods (such as Lanzcos), simpler, and oftentimes faster.

2. General matrices.

   For this environment, there exist methods that construct the low rank approximation in $O(m\,n\,\log(k))$ floating point operations.

   These methods outperform classical methods (Gram-Schmidt, Householder, *etc*) for a wide range of values of m, n, and k — often by a lot.

These methods are not like Monte Carlo.

The only similarity is in the use of randomness to address non-random problems.

When the randomized sampling method works, it is very highly accurate. Relative accuracy of 10^{-10} is typical.

The likelihood of failure is entirely negligible. It can be made as small as desired. Failure probabilities of 10^{-5} or 10^{-17} are easily obtainable. Cheap verification schemes can reduce them further.

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.*