TENSOR APPROXIMATION BY MATRIX TECHNIQUES

Eugene Tyrtyshnikov

Institute of Numerical Mathematics
Russian Academy of Sciences

HUGE-SCALE PROBLEMS

Solve numerically the integral equation

$$\int_D \frac{1}{|x-y|}\phi(y)dy = f(x), \qquad x, y \in D = [0,1]^d$$

Let $d = 3$.
Subdive the cube D into subcubes D_{ijk}:

$$D_{ijk} = [a_{i-1}, a_i] \times [a_{j-1}, a_j] \times [a_{k-1}, a_k], \qquad 0 = a_0 < a_1 < \ldots < a_n = 1$$

$$\phi(y) \approx u_{ijk} = \text{const} \quad \text{на} \quad D_{ijk} \quad \text{(collocation)} \quad \Rightarrow \quad Au = f$$

Vectors are associated with discrete functions u_{ijk}, f_{ijk} in the grid with n^3 nodes. Matrix A contains n^6 nonzero entries, none of them can be neglected, no visible structure in the case of nonuniform grids.

$n = 64 \quad \Rightarrow$ STORAGE FOR A = **512Gb** — not few!

$n = 256 \quad \Rightarrow$ STORAGE FOR A = **2Pb** (1 Pb $= 2^{50}$ byte).

A big problem is already with the storage for matrix coefficients!

CAN WE STILL SOLVE IT?

The only idea is to find a sufficiently close problem with a prominent low-parametric structure defined by reasonably few parameters, and then construct some gain-of-the-structure methods.

USE SMOOTHNESS IN MATRICES!

SMOOTHNESS = RANK STRUCTURES

SEPARATION OF VARIABLES

$$f(x_2, x_2) \approx \sum_{k=1}^{r} \phi_{1k}(x_1)\phi_{2k}(x_2)$$

$$f(x_1, x_2, ..., x_d) \approx \sum_{k=1}^{r} \phi_{1k}(x_1)\phi_{2k}(x_2)...\phi_{dk}(x_d)$$

Minimal possible r is called tensor rank.

Assume n nodes along every direction (mode). Then the number of defining parameters: $drn \ll n^d$

Separation of variables in matrices = low-rank approximation:

$$A \approx A_r = u_1 v_1^\top + ... + u_r v_r^\top$$

The number of defining parameters: $2rn \ll n^2$.

Separation of variables in multidemensional matrices
(arrays, tensors) = approximations of low tensor rank.

APPROXIMATION OF MATRICES AND 3D ARRAYS

Reshaping (reordering of multi-indices):

$$a_{ij} = a_{(i_1,i_2,i_3)(j_1,j_2,j_3)} = a_{(i_1,j_1)(i_2,j_2)(i_3,j_3)} = a_{\mathrm{ijk}}$$

$$\mathrm{i} = (i_1, j_1),\ \mathrm{j} = (i_2, j_2),\ \mathrm{k} = (i_3, j_3).$$

Tensor approximation of a matrix:

$$A \approx \tilde{A}_r = \sum_{t=1}^{r} U_t \times V_t \times W_t, \qquad U_t = [u_{(i_1,j_1)t}],\ V_t = [v_{(i_2,j_2)t}],\ W_t = [w_{(j_3,j_3)t}].$$

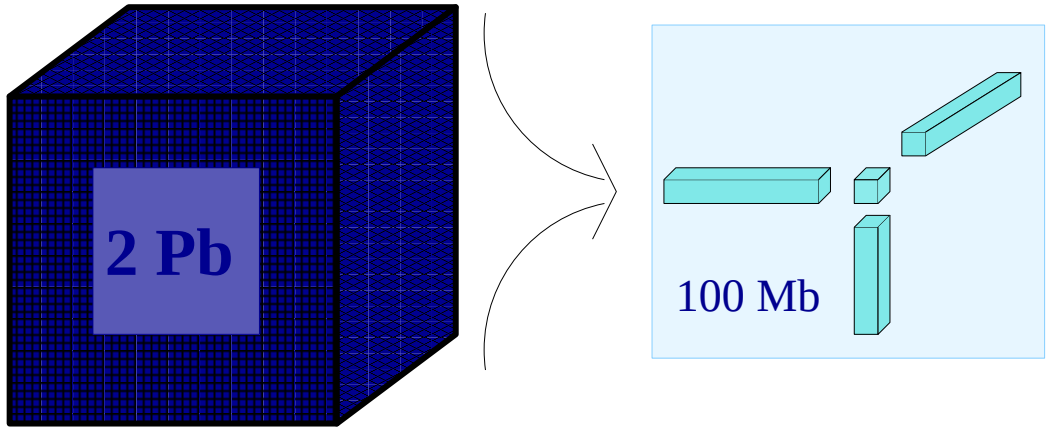
Trilinear approximation of a tensor (3D array):

$$\mathcal{A} = [a_{\mathrm{ijk}}], \qquad a_{\mathrm{ijk}} \approx \tilde{a}_{\mathrm{ijk}} = \sum_{t=1}^{r} u_{\mathrm{i}t} v_{\mathrm{j}t} w_{\mathrm{k}t}.$$

Tensor approx. of a matrix $\Leftrightarrow$ Trilinear approx. of a 3D array

IDEA FOR TENSOR COMPRESSION

SEPARATE VARIABLES!



USE ONLY SMALL PORTION OF DATA!

MAXIMAL VOLUME PRINCIPLE

Assume that

$$||\boldsymbol{A} - [\text{MATRIX OF RANK} \leq \boldsymbol{k}]||_2 \leq \boldsymbol{\varepsilon},$$

and let $\boldsymbol{A}$ be a block matrix of the form

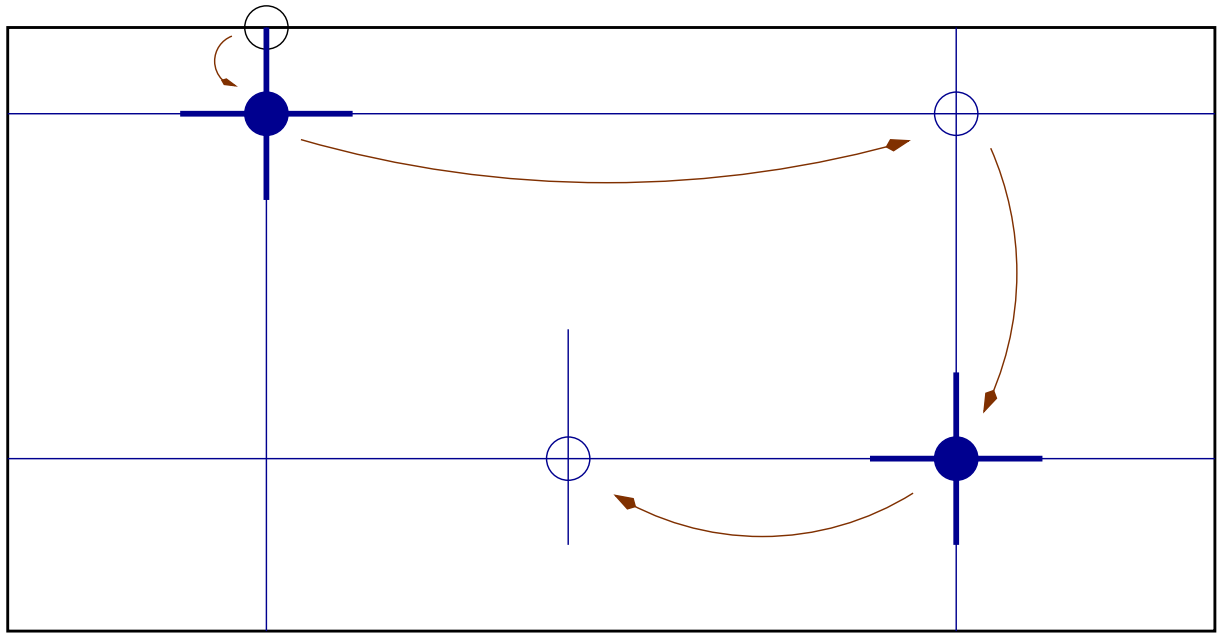
$$\begin{bmatrix} \boldsymbol{A_{11}} & \boldsymbol{A_{12}} \\ \boldsymbol{A_{21}} & \boldsymbol{A_{22}} \end{bmatrix},$$

where $\boldsymbol{A_{11}}$ is nonsingular, $\boldsymbol{k} \times \boldsymbol{k}$, and of maximal volume among all $\boldsymbol{k} \times \boldsymbol{k}$ submatrices. Then

$$||\boldsymbol{A_{22}} - \boldsymbol{A_{21}}\boldsymbol{A_{11}^{-1}}\boldsymbol{A_{12}}||_C \leq (\boldsymbol{k}+\boldsymbol{1})\ \boldsymbol{\varepsilon}.$$

- S.A.Goreinov, E.E.Tyrtyshnikov, N.L.Zamarashkin, A theory of pseudo-skeleton approximations, *Linear Algebra Appl.* 261: 1–21 (1997).
- S.A.Goreinov, E.E.Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, *Contemporary Mathematics*, Vol. 208 (2001), 47–51.

2D CROSS APPROXIMATION



Find $\boldsymbol{A} \approx \tilde{\boldsymbol{A}}_{\boldsymbol{r}} = \sum_{\boldsymbol{q}=1}^{\boldsymbol{r}} \boldsymbol{u_q} \boldsymbol{v_q}^{\top}$ (sum of *skeletons*).

0 Initialization: $\boldsymbol{p = 1}$, $\boldsymbol{j_1 = 1}$.

1 Compute column $\boldsymbol{j_p}$, subtract current approximation values $\tilde{\boldsymbol{A}}_{\boldsymbol{p}}$. Find pivot $\boldsymbol{i_p}$.

2 Compute row $\boldsymbol{i_p}$, subtract current approximation values $\tilde{\boldsymbol{A}}_{\boldsymbol{p}}$. Find pivot $\boldsymbol{j_{p+1} \neq j_p}$.

3 Using the cross $\boldsymbol{(i_p, j_p)}$, construct a new skeleton annihilating this cross.

4 Check stopping criterion, set $\boldsymbol{p := p + 1}$, return to **1**.

MILLION-SIZE COMPRESSION IN SECONDS

n	16 384	65 536	262 144	1 048 576
TIME (sec.)	0.6	2.7	14.6	61.5

Timings for the Incomplete Cross Approximation Algorithm

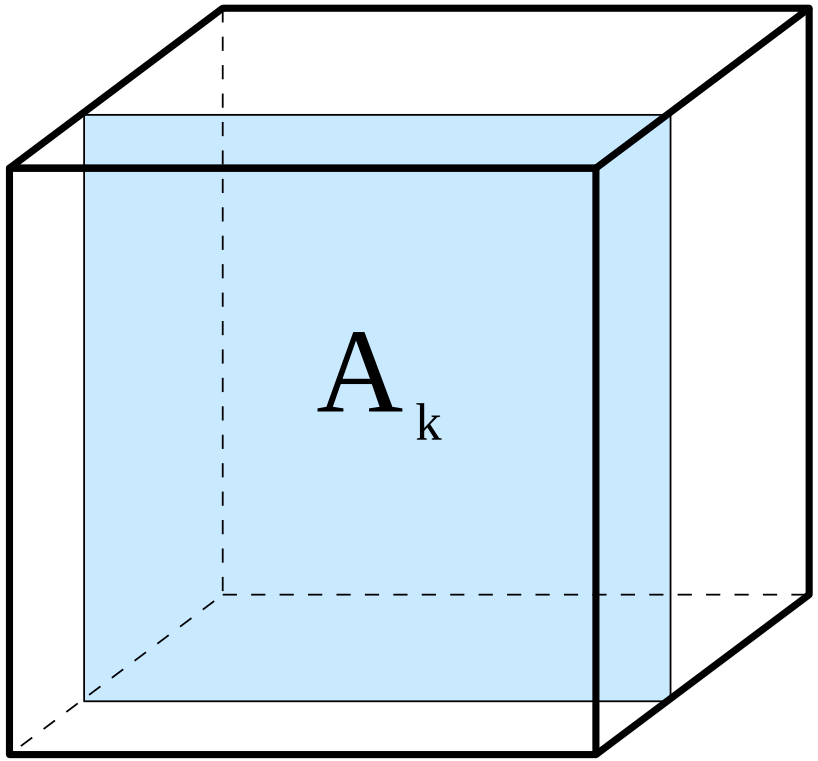
n	256	1 024	4 096	16 384	65 536
r	8	10	11	14	15
$\tilde{\varepsilon}$	$\mathbf{3.3 \cdot 10^{-6}}$	$\mathbf{3.3 \cdot 10^{-6}}$	$\mathbf{3.3 \cdot 10^{-6}}$	$\mathbf{1.8 \cdot 10^{-6}}$	$\mathbf{6.5 \cdot 10^{-6}}$
$\|\|A-B\|\|_F/\|\|A\|\|_F$	$\mathbf{2.9 \cdot 10^{-6}}$	$\mathbf{2.6 \cdot 10^{-6}}$	$\mathbf{6.4 \cdot 10^{-6}}$	$\mathbf{2.2 \cdot 10^{-6}}$	$\mathbf{4.7 \cdot 10^{-6}}$

Verification of the Incomplete Cross Approximation Algorithm

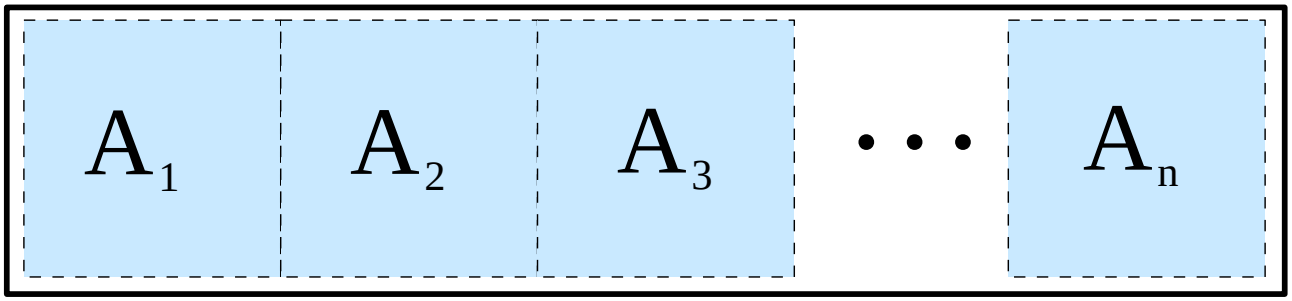
- S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin: A Theory of Pseudo-Skeleton Approximations, *Linear Algebra Appl.* 261: 1–21, 1997.
- E. E. Tyrtyshnikov: Incomplete cross approximation in the mosaic-skeleton method, *Computing* 64, no. 4 (2000), 367–380.
- S. A. Goreinov, E. E. Tyrtyshnikov: The maximal-volume concept in approximation by low-rank matrices, *Contemporary Mathematics*, Vol. 208, 2001, 47-51.

3D ARRAYS AND MATRICES

Slices in one mode



Matrices of slices



TRILINEAR DECOMPOSITION

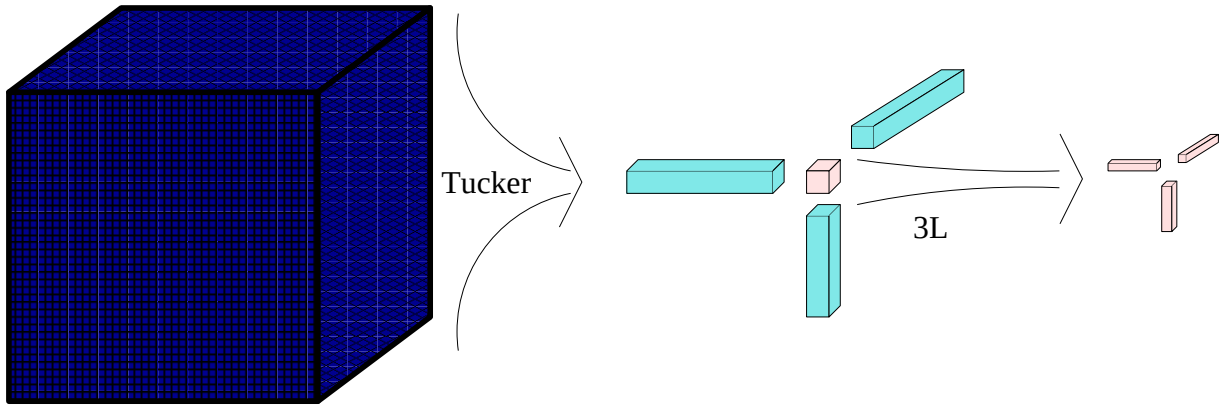
All methods are too slow
for $\boldsymbol{n \geq 128.}$

TUCKER DECOMPOSITION

$$\boldsymbol{a_{ijk} = \sum_{i'=1}^{r_1} \sum_{j'=1}^{r_2} \sum_{k'=1}^{r_3} g_{i'j'k'} u_{ii'} v_{jj'} w_{kk'},}$$

Tucker factors $\boldsymbol{U, V, W}$ are orthogonal matrices,
array $\boldsymbol{\mathcal{G} = [g_{i'j'k'}]}$ is much smaller than $\boldsymbol{\mathcal{A}}$.

TRILINEAR DECOMPOSITION FOR LARGE $\boldsymbol{n}$



TUCKER DECOMPOSITION

1. Consider matrices of slices

$$\begin{aligned} \boldsymbol{A}^{(1)} &= [\boldsymbol{a}^{1}_{i(jk)}] = [\boldsymbol{a}_{ijk}], \\ \boldsymbol{A}^{(2)} &= [\boldsymbol{a}^{2}_{j(ki)}] = [\boldsymbol{a}_{ijk}], \\ \boldsymbol{A}^{(3)} &= [\boldsymbol{a}^{3}_{k(ij)}] = [\boldsymbol{a}_{ijk}], \end{aligned}$$

2. Compute $\boldsymbol{SVD}$ for each of them.

$$\boldsymbol{A}^{(1)} = \boldsymbol{U}\boldsymbol{\Sigma}_1\boldsymbol{\Phi}_1^\top, \qquad \boldsymbol{A}^{(2)} = \boldsymbol{V}\boldsymbol{\Sigma}_2\boldsymbol{\Phi}_2^\top, \qquad \boldsymbol{A}^{(3)} = \boldsymbol{W}\boldsymbol{\Sigma}_3\boldsymbol{\Phi}_3^\top,$$

3. Find the Tucker core via transformation

$$\boldsymbol{g}_{i'j'k'} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \boldsymbol{a}_{ijk}\boldsymbol{u}_{ii'}\boldsymbol{v}_{jj'}\boldsymbol{w}_{kk'}.$$

Complexity = $\mathcal{O}(n^4)$ plus n^3 computations of the entries of $\mathcal{A}$.

We suggest an algorithm with almost linear complexity

Complexity = $\mathcal{O}(nr^3)$ plus $\mathcal{O}(nr^2)$ computations of the entries of $\mathcal{A}$.

3D CROSS EXISTENCE THEOREM

Suppose we are aware that

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_i \boldsymbol{U} \times_j \boldsymbol{V} \times_k \boldsymbol{W} + \boldsymbol{\mathcal{E}}, \quad ||\boldsymbol{\mathcal{E}}|| = \varepsilon$$

holds for some $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ and $\boldsymbol{\mathcal{G}}$. Then there exist matrices $\boldsymbol{U'}, \boldsymbol{V'}$ and $\boldsymbol{W'}$ of sizes $\boldsymbol{n_1} \times \boldsymbol{r_1}$, $\boldsymbol{n_2} \times \boldsymbol{r_2}$ and $\boldsymbol{n_3} \times \boldsymbol{r_3}$ and consisting of some $\boldsymbol{r_1}$ columns, $\boldsymbol{r_2}$ rows and $\boldsymbol{r_3}$ fibers, of $\boldsymbol{\mathcal{A}}$, respectively, and a tensor $\boldsymbol{\mathcal{G}'}$ such that

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}'} \times_i \boldsymbol{U'} \times_j \boldsymbol{V'} \times_k \boldsymbol{W'} + \boldsymbol{\mathcal{E}'},$$

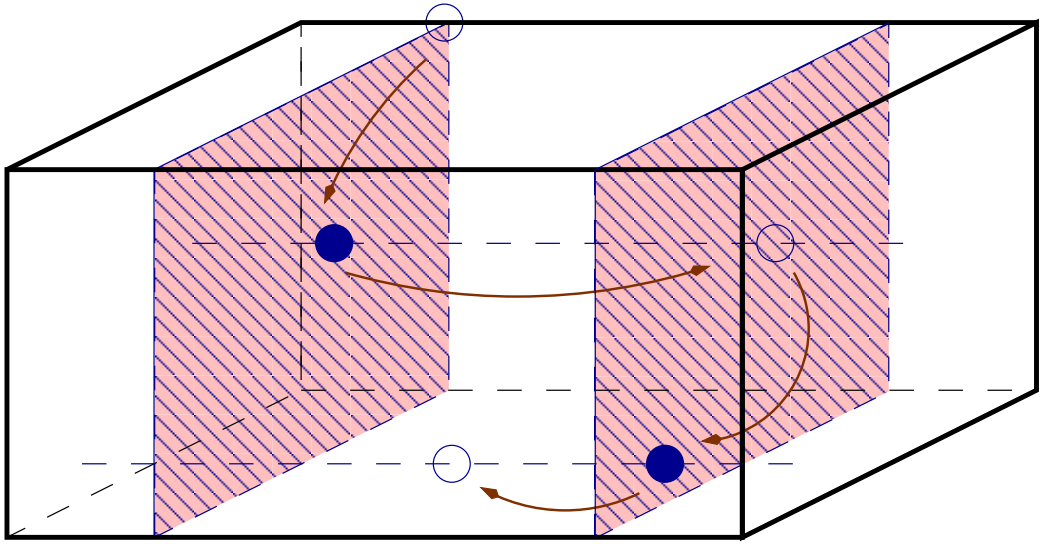
where

$$||\boldsymbol{\mathcal{E}'}||_C \leq (\boldsymbol{r_1 r_2 r_3 + 2 r_1 r_2 + 2 r_1 + 1})\boldsymbol{\varepsilon}.$$

I.Oseledets, D.Savostyanov, E.Tyrtyshnikov,
Tucker dimensionality reduction of three-dimensional arrays in linear time,
submitted to SIMAX, 2006.

3D CROSS APPROXIMATION

Complexity = $\mathcal{O}(n^2)$



Find $\mathcal{A} \approx \tilde{\mathcal{A}} = \sum_{q=1}^{r} A_q \times w_q$.

1 Compute the slice A_{k_p}, subtract approx. values $\tilde{\mathcal{A}}$.
Find pivot A_{k_p}.

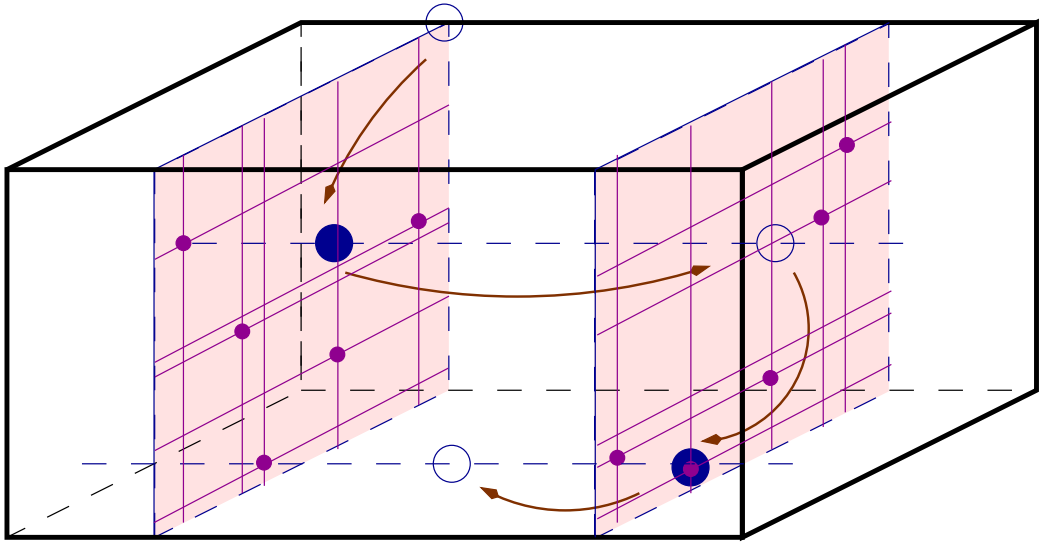
2 Compute a fiber w_p, subtract approx. values $\tilde{\mathcal{A}}$. Find pivot $k_{p+1} \neq k_p$.

3 Skeleton $A_{k_p} \times w_p$ nullifies the slice-by-fiber cross.

4 Check stopping criterion, set $p := p + 1$, return to **1**.

3D CROSS APPROXIMATION

Complexity $= \mathcal{O}(nr^4)$



Find $\boldsymbol{\mathcal{A} \approx \tilde{\mathcal{A}} = \sum_{q=1}^{r^2} u_q \times v_q \times w_q.}$

1 Compute a *cross approximation* of the slice $\boldsymbol{A_{k_p} = \sum_{q=1}^{r} u_{pq} v_{pq}^{\top},}$

subtract approx. values $\boldsymbol{\tilde{\mathcal{A}}}$.

Find pivot $\boldsymbol{A_{k_p}}$ (HOW CAN WE DO THIS?)

2 Compute a fiber $\boldsymbol{w_p}$, subtract approx. values $\boldsymbol{\tilde{\mathcal{A}}}$. Find pivot $\boldsymbol{k_{p+1} \neq k_p}$.

3 Skeleton $\sum^{r} \boldsymbol{u_{pq} \times v_{pq} \times w_p}$ nullifies the slice-by-fiber cross.

4 Check stopping criterion, set $\boldsymbol{p := p + 1}$, return to **1**.

NUMERICAL RESULTS

$$a_{ijk} = 1/\sqrt{i^2 + j^2 + k^2}, \qquad 1 \le i, j, k \le n$$

Ranks and approximation accuracy

	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	7	$3.77_{10}-4$	11	$3.91_{10}-6$	14	$5.7_{10}-8$	18	$2.21_{10}-10$
128	8	$5.19_{10}-4$	12	$5.92_{10}-6$	17	$2.00_{10}-8$	20	$5.63_{10}-10$
256	9	$4.11_{10}-4$	14	$6.4_{10}-6$	19	$3.46_{10}-8$	23	$4.5_{10}-10$
512	9	$4.93_{10}-4$	15	$6.67_{10}-6$	21	$2.92_{10}-8$	26	$3.27_{10}-10$
1024	10	$5.47_{10}-4$	17	$3.21_{10}-6$	23	$3.95_{10}-8$	29	$4.73_{10}-10$
2048	11	$4.98_{10}-4$	18	$5.26_{10}-6$	25	$6.83_{10}-8$	31	$5.94_{10}-10$
4096	11	$8.4_{10}-4$	19	$4.25_{10}-6$	27	$3.56_{10}-8$	34	$3.38_{10}-10$
8192	12	$6.8_{10}-4$	20	$6.00_{10}-6$	28	$5.8_{10}-8$	36	$3.66_{10}-10$
16384	13	$2.69_{10}-4$	22	$4.78_{10}-6$	30	$5.65_{10}-8$	39	$2.67_{10}-10$
32768	13	$8.52_{10}-4$	23	$6.09_{10}-6$	32	$7.16_{10}-8$	41	$5.51_{10}-10$
65536	14	$6.27_{10}-4$	24	$6.52_{10}-6$	34	$7.89_{10}-8$	43	$1.41_{10}-9$

NUMERICAL RESULTS

$$a_{ijk} = 1/\sqrt{i^2 + j^2 + k^2}, \qquad 1 \le i, j, k \le n$$

Ranks and memory savings

		1.10^{-3}		1.10^{-5}		1.10^{-7}		1.10^{-9}	
n	full	r	mem	r	mem	r	mem	r	mem
64	2Mb	7		11		14		18	
128	16Mb	8		12		17		20	
256	128Mb	9		14		19		23	
512	1Gb	9		15		21		26	
1024	8Gb	10		17		23		29	
2048	64Gb	11		18		25		31	
4096	512Gb	11		19		27		34	
8192	4Tb	12	2.5Mb	20	4Mb	28	5.2Mb	36	7Mb
16384	32Tb	13	5Mb	22	8Mb	30	11Mb	39	15Mb
32768	256Tb	13	10Mb	23	17Mb	32	24Mb	41	20Mb
65536	2Pb	14	21Mb	24	36Mb	34	51Mb	43	64Mb

NUMERICAL RESULTS

$$a_{ijk} = 1/(i^2 + j^2 + k^2), \qquad 1 \le i, j, k \le n$$

Ranks and approximation accuracy

	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	8	$3.43_{10}-4$	12	$2.18_{10}-6$	15	$4.1_{10}-8$	18	$5.63_{10}-10$
128	9	$4.25_{10}-4$	13	$5.81_{10}-6$	18	$2.06_{10}-8$	21	$5.26_{10}-10$
256	10	$4.4_{10}-4$	15	$3.89_{10}-6$	20	$2.86_{10}-8$	24	$4.78_{10}-10$
512	11	$4.07_{10}-4$	17	$3.49_{10}-6$	22	$3.78_{10}-8$	27	$4.55_{10}-10$
1024	12	$4.78_{10}-4$	18	$6.27_{10}-6$	24	$5.39_{10}-8$	30	$3.7_{10}-10$
2048	12	$4.05_{10}-4$	20	$3.73_{10}-6$	26	$6.21_{10}-8$	33	$3.31_{10}-10$
4096	13	$3.8_{10}-4$	21	$5.24_{10}-6$	28	$5.11_{10}-8$	36	$2.37_{10}-10$
8192	14	$6.14_{10}-4$	22	$4.56_{10}-6$	31	$2.85_{10}-8$	38	$3.78_{10}-10$
16384	15	$8.08_{10}-4$	24	$4.19_{10}-6$	32	$4._{10}-8$	41	$5.65_{10}-10$
32768	15	$8.2_{10}-4$	25	$4.66_{10}-6$	34	$5.41_{10}-8$	44	$2.4_{10}-10$
65536	16	$2.98_{10}-4$	26	$5.69_{10}-6$	36	$6.46_{10}-8$	46	$4.38_{10}-10$

NUMERICAL RESULTS

$$a_{ijk} = 1/(i^2 + j^2 + k^2), \qquad 1 \le i, j, k \le n$$

Ranks and memory savings

		$1\cdot10^{-3}$		$1\cdot10^{-5}$		$1\cdot10^{-7}$		$1\cdot10^{-9}$	
n	full	r	mem	r	mem	r	mem	r	mem
64	2Mb	8		12		15		18	
128	16Mb	9		13		18		21	
256	128Mb	10		15		20		24	
512	1Gb	11		17		22		27	
1024	8Gb	12		18		24		30	
2048	64Gb	12		20		26		33	
4096	512Gb	13		21		28		36	
8192	4Tb	14		22		31		38	
16384	32Tb	15	5Mb	24	9Mb	32	12Mb	41	15Mb
32768	256Tb	15	11Mb	25	19Mb	34	26Mb	44	33Mb
65536	2Pb	16	24Mb	26	40Mb	36	54Mb	46	69Mb

NUMERICAL RESULTS

$$a_{ijk} = 1/(i^2 + j^2 + k^2)^{3/2}, \qquad 1 \le i, j, k \le n$$

Ranks and approximation accuracy

	$1._{10}-3$		$1._{10}-5$		$1._{10}-7$		$1._{10}-9$	
n	r	err	r	err	r	err	r	err
64	7	$3.81_{10}-4$	11	$3.45_{10}-6$	15	$2.03_{10}-8$	18	$2.54_{10}-10$
128	8	$2.95_{10}-4$	12	$5.37_{10}-6$	16	$5.36_{10}-8$	20	$5.59_{10}-10$
256	8	$3.82_{10}-4$	13	$6.68_{10}-6$	18	$6.09_{10}-8$	23	$2.17_{10}-10$
512	8	$3.56_{10}-4$	14	$3.96_{10}-6$	20	$3.77_{10}-8$	25	$4.26_{10}-10$
1024	8	$3.73_{10}-4$	15	$3.92_{10}-6$	21	$4.66_{10}-8$	27	$3.68_{10}-10$
2048	8	$3.72_{10}-4$	16	$2.21_{10}-6$	23	$2.58_{10}-8$	29	$4.81_{10}-10$
4096	8	$3.74_{10}-4$	16	$3.84_{10}-6$	24	$2.5_{10}-8$	31	$4.53_{10}-10$
8192	8	$3.74_{10}-4$	16	$4.14_{10}-6$	25	$4.92_{10}-8$	32	$1.02_{10}-9$
16384	8	$3.76_{10}-4$	16	$6.16_{10}-6$	25	$5.14_{10}-8$	34	$9.38_{10}-10$
32768	8	$3.75_{10}-4$	16	$4.82_{10}-6$	26	$5.46_{10}-8$	36	$3.45_{10}-10$
65536	8	$3.75_{10}-4$	16	$9.00_{10}-6$	26	$7.78_{10}-8$	37	$5.28_{10}-10$

THEORY: TENSOR RANK ESTIMATES

On *almost uniform* grids $\boldsymbol{h^{-1} \sim n}$

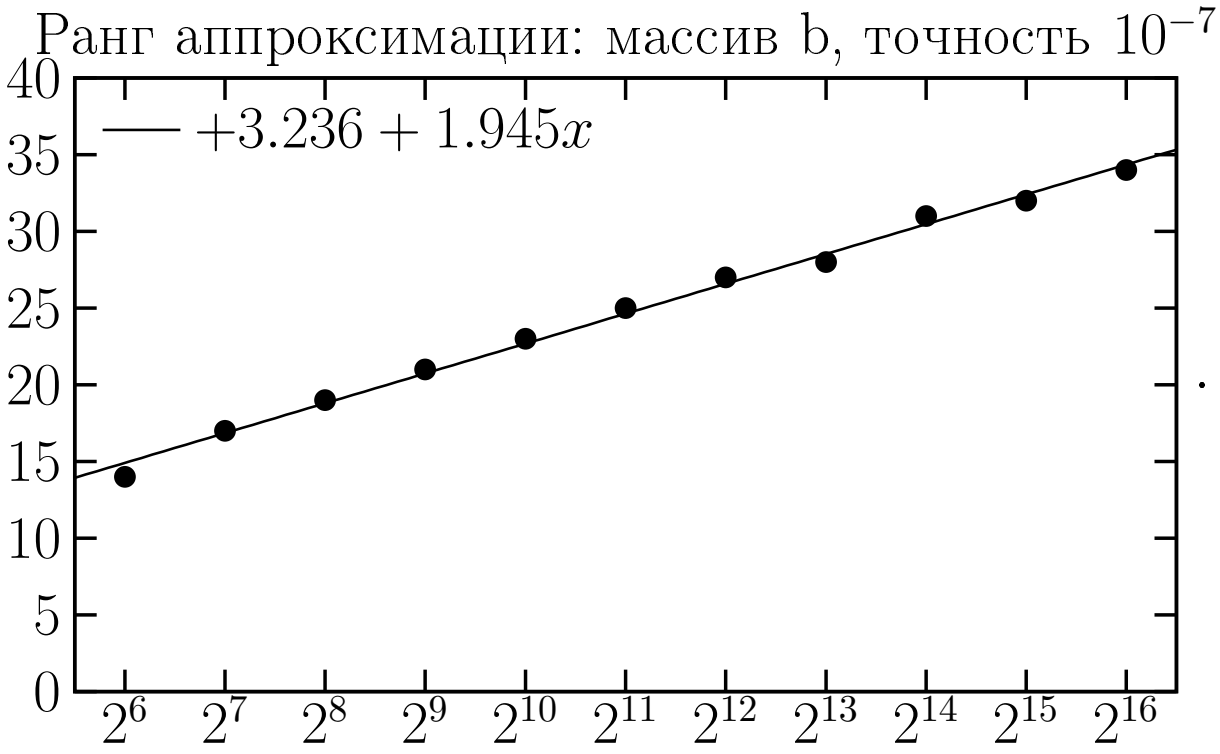
$$\boldsymbol{r \leq c \log n \log^2 \varepsilon^{-1}.}$$

E.E.Tyrtyshnikov,
Tensor approximations of matrices generated by asymptotically smooth functions, *Sbornik: Mathematics* **194**, No. 5-6 (2003), 941–954
(translated from *Mat. Sb.* **194**, No. 6 (2003), 146–160).

PRACTICAL PROOF

$$a_{ijk} = 1/\sqrt{i^2 + j^2 + k^2}, \qquad 1 \le i, j, k \le n$$

Tensor rank versus array size

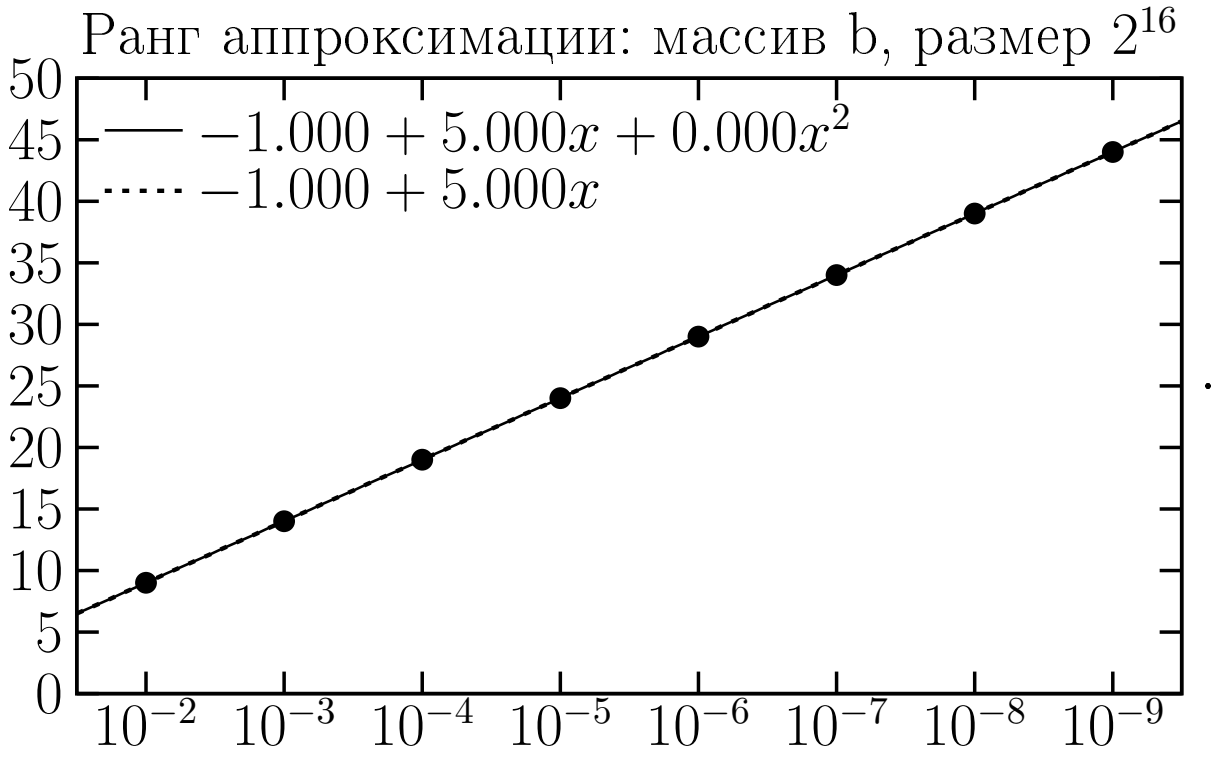


$$r \sim \log n$$

PRACTICAL PROOF

$$a_{ijk} = 1/\sqrt{i^2 + j^2 + k^2}, \qquad 1 \le i, j, k \le n$$

Tensor rank versus approximation error

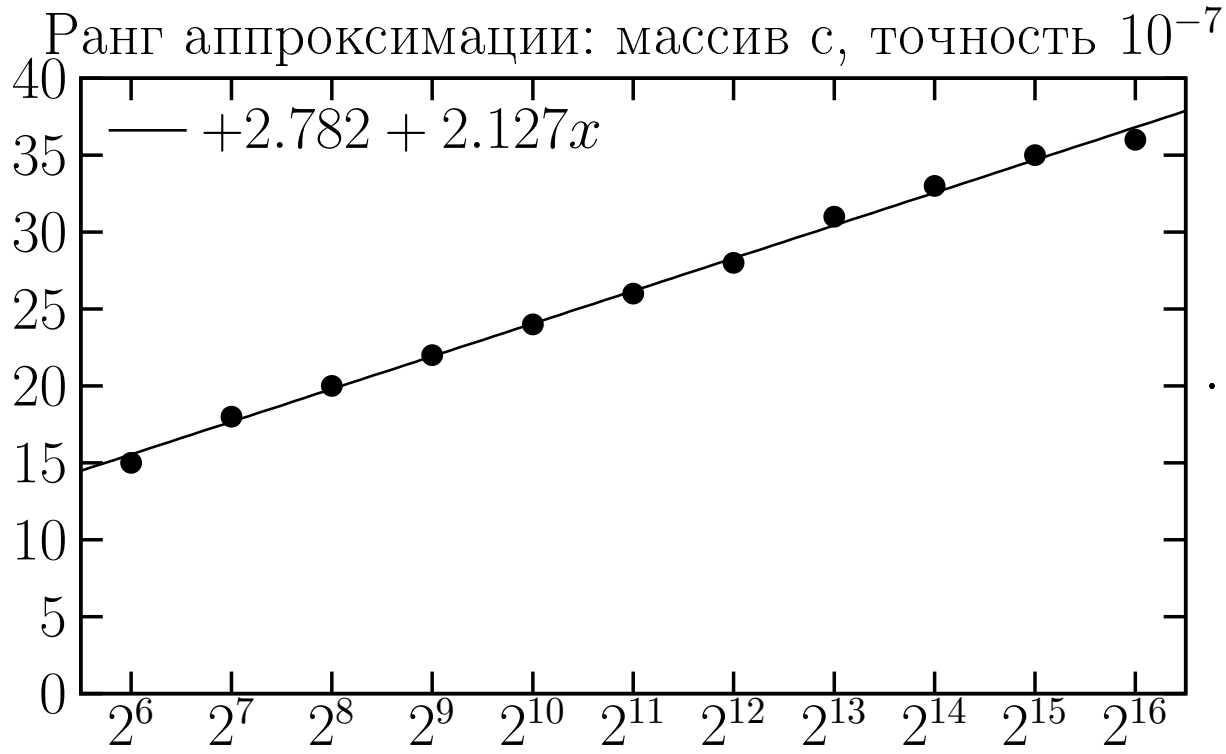


$$r \sim \log \varepsilon^{-1}$$

PRACTICAL PROOF

$$a_{ijk} = 1/(i^2 + j^2 + k^2), \qquad 1 \le i, j, k \le n$$

Tensor rank versus array size

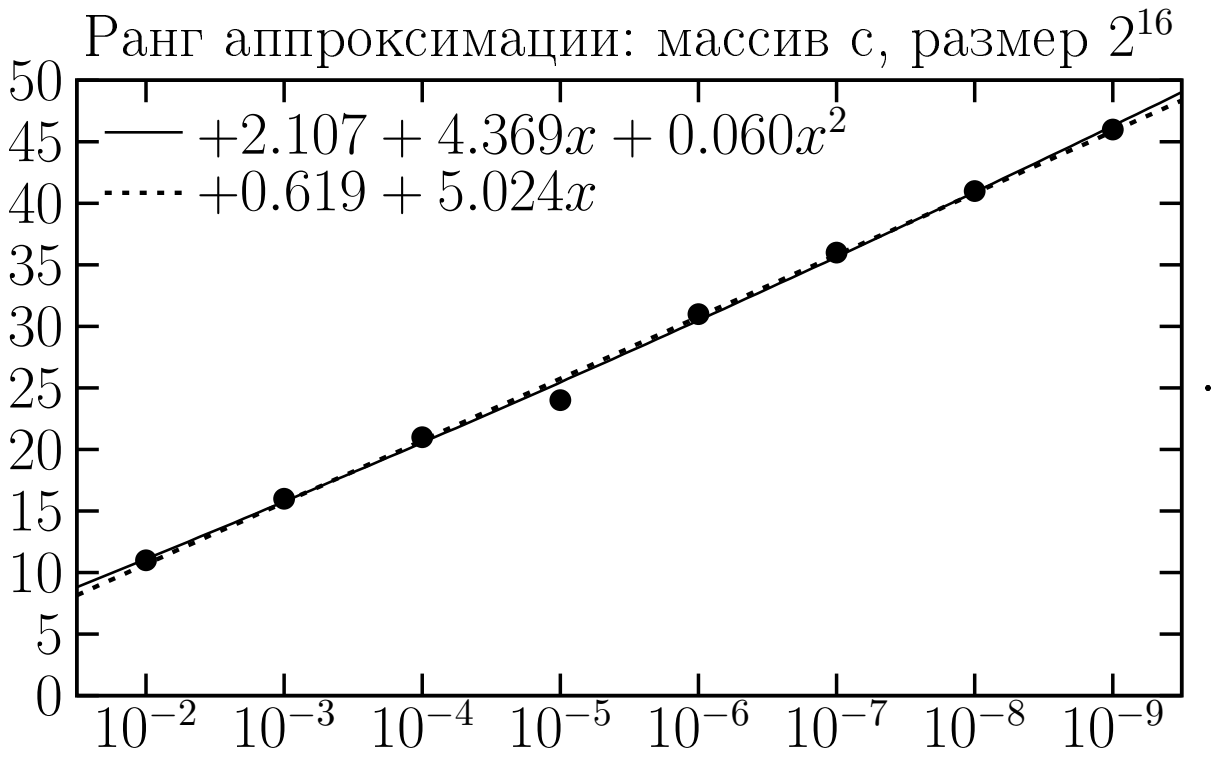


$$r \sim \log n$$

PRACTICAL PROOF

$$a_{ijk} = 1/(i^2 + j^2 + k^2), \qquad 1 \le i, j, k \le n$$

Tensor rank versus approximation error



$$r \sim \log \varepsilon^{-1}$$

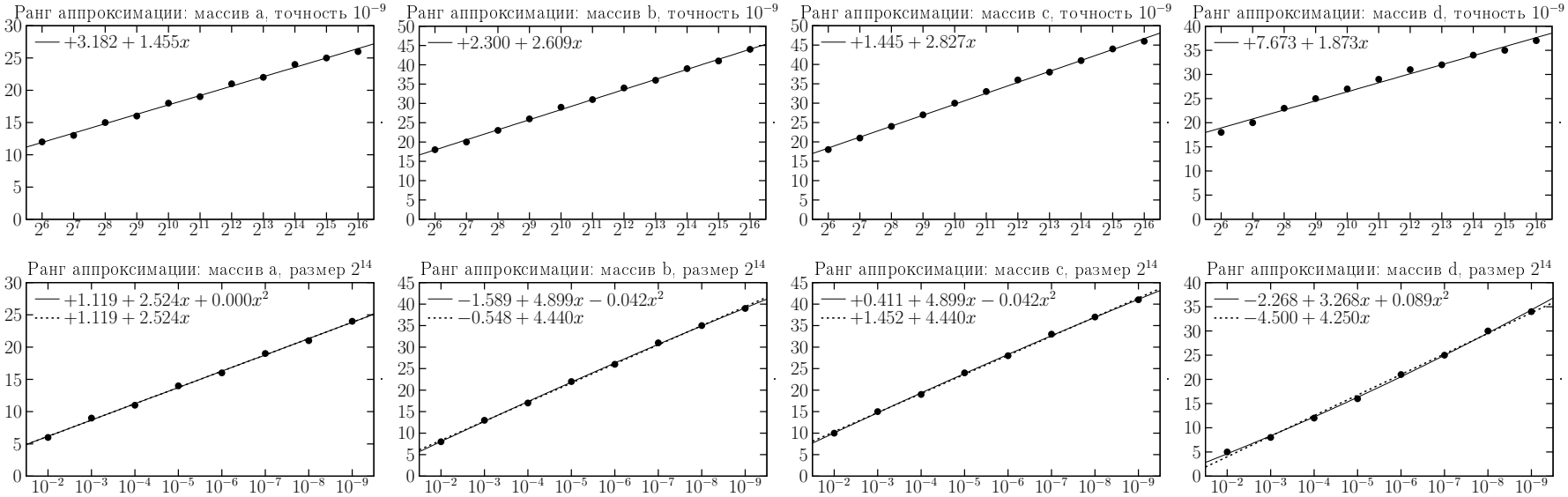
ASYMPTOTICS OF TENSOR RANK

Theory

$$r \lesssim \log n \log^2 \varepsilon^{-1}$$

Practice

$$r \sim \log n \log \varepsilon^{-1}$$



3D TENSOR SOLVER WITH TENSOR VECTORS

$$\int_{\boldsymbol{D}} \frac{\mathbf{1}}{|\boldsymbol{x}-\boldsymbol{y}|}\boldsymbol{\phi}(\boldsymbol{y})\boldsymbol{dy} = \boldsymbol{f}(\boldsymbol{x}), \qquad \boldsymbol{x},\boldsymbol{y} \in \boldsymbol{D} = [\mathbf{0}:\mathbf{1}]^{\mathbf{3}}$$

$$\boldsymbol{Au} = \boldsymbol{f}, \qquad \boldsymbol{u_{ijk}}, \boldsymbol{f_{ijk}} \quad \text{on the grid} \quad \boldsymbol{n} \times \boldsymbol{n} \times \boldsymbol{n}.$$

grid size $\boldsymbol{n}$	**16**	**32**	**64**	**128**	**256**	**512**
full matrix	**128**Mb	**8**Gb	**512**Gb	**32**Tb	**2**Pb	**128**Pb
tensor format	**50**Kb	**200**Kb	**1.1**Mb	**5**Mb	**22**Mb	**96**Mb
time	**0.3**sec	**1.5**sec	**12.4**sec	**48**sec	**2.5**min	**16**min

UNIFORM 1D GRIDS $\Rightarrow$ 3 LEVEL TOEPLITZ MATRICES

DOES IT MAKE THINGS EASIER?

Could be so, but only if Toeplitz property
is combined with some rank structures!

TOEPLITZ AND RANK STRUCTURES

If $\boldsymbol{d = 1}$ and grid is unform then

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{1-n} \\ a_1 & a_0 & a_{-1} & \cdots & a_{2-n} \\ a_2 & a_1 & a_0 & \cdots & a_{3-n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{bmatrix} \quad \text{(Toeplitz matrix)}$$

$$Z_s = \begin{bmatrix} 0 & \ldots & 0 & s \\ 1 & & & 0 \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix} \quad \Rightarrow \quad \mathbf{rank}(A - Z_s A Z_t^\top) \leq 2$$

Operator $\boldsymbol{A \mapsto A - Z_s A Z_t^\top = G H^\top}$ is invertible for $\boldsymbol{st \neq 1}$:

$$\boldsymbol{(1-st)A = \sum_{k=1}^{r} C_s(g_k) C_t^\top(h_k), \qquad G = [g_1, ..., g_r], \quad H = [h_1, ..., h_r]}$$

$$\boldsymbol{[C_s(v)]_{ij} = \begin{cases} v_{i-j}, & i - j \geq 0, \\ s\, v_{n+i-j}, & i - j < 0. \end{cases}} \qquad (\boldsymbol{s}\text{-circulant})$$

FORMULAS FOR THE INVERSE MATRICES

$$\mathbf{rank}(\boldsymbol{A - Z_s A Z_t^\top}) = \mathbf{rank}(\boldsymbol{A^{-1} - Z_t^\top A^{-1} Z_s}) = \mathbf{rank}(\boldsymbol{A^{-\top} - Z_t A^{-\top} Z_s^\top})$$

Trench, Gohberg–Sementsul, Krupnik, Sahnovich, Heinig, Kailath, ...

Gohberg–Sementsul:

$$A^{-1} = x_0^{-1} \begin{bmatrix} x_0 & & & \\ x_1 & x_0 & & \\ \dots & \dots & \dots & \\ x_n & \dots & \dots & x_0 \end{bmatrix} \begin{bmatrix} u_0 & u_1 & \dots & u_n \\ & u_0 & \dots & u_{n-1} \\ & & \dots & \dots \\ & & & u_0 \end{bmatrix} -$$

$$- x_0^{-1} \begin{bmatrix} 0 & & & & \\ y_0 & 0 & & & \\ y_1 & y_0 & 0 & & \\ \dots & \dots & \dots & \dots & \\ y_{n-1} & \dots & \dots & y_0 & 0 \end{bmatrix} \begin{bmatrix} 0 & v_0 & v_1 & \dots & v_{n-1} \\ & 0 & v_0 & \dots & v_{n-2} \\ & & \dots & \dots & \dots \\ & & & 0 & v_0 \\ & & & & 0 \end{bmatrix}$$

$$A \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \dots \\ 0 \end{bmatrix}, \quad A \begin{bmatrix} y_0 \\ \dots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} 0 \\ \dots \\ 0 \\ 1 \end{bmatrix}, \quad u_i = y_{n-i}, \quad w_i = x_{n-1}, \qquad 0 \le i \le n.$$

A CHALLENGE OF 2D TOEPLITZ MATRICES

Two-level Toeplitz matrix of order $n = p^2$:

$$A = [a_{k_1 - l_1}], \quad 0 \le k_1, l_1 \le p - 1,$$

$$a_{k_1 - l_1} = [a_{k_1 - l_1, k_2 - l_2}], \qquad 0 \le k_2, l_2 \le p - 1.$$

A can be viewed as a block Toeplitz matrix with $p \times p$ blocks $\Rightarrow$ Gohberg-Heinig formulas for A^{-1}.
But too many parameters: $O(n^{3/2})$.

PROPOSAL

Write (or approximate)

$$A = A_1 \otimes B_1 + \ldots + A_r \otimes B_r$$

with Toeplitz A_i, B_i and $r \ll n$.

If a 2-level Toeplitz matrix is in the form

$$\boldsymbol{A = A_1 \otimes B_1 + ... + A_r \otimes B_r}$$

then we may expect that

$$\boldsymbol{A^{-1} \approx L_1 \otimes U_1 + ... + L_\rho \otimes U_\rho}$$

with $\boldsymbol{\rho = \rho(r, n, \varepsilon) \ll n}$.
But $\boldsymbol{L_i}$, $\boldsymbol{U_i}$ do not inherit Toeplitz property.

Maybe $\boldsymbol{L_i}$, $\boldsymbol{U_i}$ can be of low displacement rank
(a sum or products of Toeplitz matrices)?

SUBLINEAR COMPLEXITY FOR 2D TOEPLITZ INVERSION

Consider 2-level Toeplitz matrices of order $N = n^2$:
block Toeplitz with Toeplitz blocks of order n.

Example: 5-point Laplacian. The inverse matrix is approximated by the Newton–Schultz method

$$X_{k+1} = \texttt{APPROXIMATION}(2X_k - X_k A X_k)$$

with a rank-structured approxination of all computed matrices: by matrices of limited tensor rank and limited displacemnet rank if each block.

n	64^2	128^2	256^2	512^2
Tensor rank A^{-1}	9	10	11	12
Averaged displacement rank of A^{-1}	13.5	13.5	16.8	18.6

Inversion of the 5-point Laplacian

Time behaves as $\mathcal{O}(\sqrt{N} r_{\mathbf{mean}}^2)$, where $r_{\mathbf{mean}}$ is the averaged displacement rank.

Low-parametric representations of inverse matrices contain $o(N)$ parameters. Hence, all difficulties are relegated to the representation of vectors, not matrices!

A BETTER PROPOSAL

Approximate $\boldsymbol{L_i}$ and $\boldsymbol{U_i}$ by
CIRCULANT PLUS LOW-RANK!

THEOREM.

$$\texttt{tensor rank}(\boldsymbol{I} + \boldsymbol{A} \otimes \boldsymbol{u}\boldsymbol{v}^\top + \boldsymbol{p}\boldsymbol{q}^\top \otimes \boldsymbol{B})^{-1} \leq \boldsymbol{7}$$

TENSOR DECOMPOSITION
= SIMULTANEOUS DIAGONALIZATION

$$a_{ijk} = \sum_{s=1}^{r} u_{is} v_{js} w_{ks}, \quad 1 \le i, j \le n, \quad 1 \le k \le r$$

$$A_k = [a_{ijl}], \quad 1 \le i, j \le n$$

$$A_k = U \begin{bmatrix} w_{k1} & & \\ & \ddots & \\ & & w_{kr} \end{bmatrix} V^\top, \qquad U = [u_{is}],\ V = [v_{js}]$$

If $\boldsymbol{A_1, ..., A_r}$ are triangular matrices, the approximate simultaneous diagonalization is not a big deal. So try to reduce the problem to triangular matrices! With the least squares, best suitable is ORTHOGONAL reduction.

FAST SIMULTANEOUS ORTHOGONAL REDUCTION TO TRIANGULAR MATRICES

Given $\boldsymbol{n \times n}$ *real matrices* $\boldsymbol{A_1, ..., A_r}$, *find orthogonal* $\boldsymbol{n \times n}$ *matrices* $\boldsymbol{Q}$ *and* $\boldsymbol{Z}$ *such that matrices*

$$\boldsymbol{B_k = Q A_k Z}$$

are as upper triangular as possible.

- I.Oseledets, D.Savostyanov, E.Tyrtyshnikov,
  Fast simultaneous orthogonal reduction to triangular matrices,
  submitted to SIMAX, 2006.

SIMULTANEOUS EIGENVALUE PROBLEM

Given real matrices $\boldsymbol{A_1}$, ..., $\boldsymbol{A_r}$, *find orthogonal* $\boldsymbol{Q}$ *and* $\boldsymbol{Z}$ *making matrices* $\boldsymbol{QA_1Z}$, ..., $\boldsymbol{QA_rZ}$ *as upper triangular as possible.*

DEFLATION STEP:

$$\boldsymbol{QA_kZ} \approx \begin{pmatrix} \boldsymbol{\lambda_k} & \boldsymbol{v_k^\top} \\ \boldsymbol{0} & \boldsymbol{B_k} \end{pmatrix} \qquad \Leftrightarrow \qquad \boldsymbol{QA_kZe_1} \approx \boldsymbol{\lambda_k e_1}$$

$$\boldsymbol{A_k x = \lambda_k y}, \qquad \boldsymbol{x = Ze_1}, \quad \boldsymbol{y = Q^\top e_1}.$$

ALGORITHM. Given r real matrices $A_1, \ldots, A_r$ of size $n \times n$, find orthogonal matrices Q and Z such that the matrices QA_kZ are as upper triangular as possible:

1. Set $m = n,\ B_i = A_i,\ i = 1, \ldots, r,\ Q = Z = I.$
2. If m = 1 then stop.
3. Solve the simultaneous eigenvalue problem $B_k x = \lambda_k y,\ k = 1, \ldots, r.$
4. Find $m \times m$ Householder matrices Q_m, Z_m such that
$$x = \alpha_1 Q_m^\top e_1,\ y = \alpha_2 Z_m e_1.$$
5. Calculate C_k as $(m-1) \times (m-1)$ submatrices of matrices $\widehat{B}_k$ defined as follows:
$$\widehat{B}_k = QB_kZ = \begin{pmatrix} \alpha_k & v_k^\top \\ \varepsilon_k & C_k \end{pmatrix}.$$
6. Set
$$Q \leftarrow \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Q_m \end{pmatrix} Q,\ Z \leftarrow Z \begin{pmatrix} I_{(n-m)\times(n-m)} & 0 \\ 0 & Z_m \end{pmatrix}.$$
7. Set $m = m - 1,\ B_k = C_k$ and proceed to the step 2.

REFORMULATION OF SEP

$$\sum_{j=1}^{n} (\boldsymbol{A}_k)_{ij} \boldsymbol{x}_j = \boldsymbol{\lambda}_k \boldsymbol{y}_i. \tag{1}$$

Introduce $\boldsymbol{r} \times \boldsymbol{n}$ matrices $\boldsymbol{B_1}, ..., \boldsymbol{B_r}$:

$$(\boldsymbol{B}_j)_{ki} = \boldsymbol{a}_{ijk} = (\boldsymbol{A}_k)_{ij}, \quad \boldsymbol{k} = \boldsymbol{1}, ..., \boldsymbol{r}, \quad \boldsymbol{i} = \boldsymbol{1}, ..., \boldsymbol{n}, \quad \boldsymbol{j} = \boldsymbol{1}, ..., \boldsymbol{n},$$

and a column vector $\boldsymbol{\lambda} = [\boldsymbol{\lambda_1}, \ ..., \ \boldsymbol{\lambda_r}]^\top$.

SEP:

$$\sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{B}_j = \boldsymbol{\lambda} \boldsymbol{y}^\top = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ ... \\ \boldsymbol{\lambda}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 & ... & \boldsymbol{y}_n \end{bmatrix}.$$

Find a linear combination of the given matrices to produce a rank-1 matrix.

Gauss-Newton algorithm for the simultaneous eigenvalue problem

Linearize the system

$$\sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{B}_j = \boldsymbol{\lambda} \boldsymbol{y}^\top = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\ldots} \\ \boldsymbol{\lambda}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{\ldots} & \boldsymbol{y}_n \end{bmatrix}$$

and solve the obtained overdetermined linear system

$$\sum_{j=1}^{n} \widehat{\boldsymbol{x}}_j \boldsymbol{B}_j = \triangle \boldsymbol{\lambda} \boldsymbol{y}^\top + \boldsymbol{\lambda} \triangle \boldsymbol{y}^\top, \quad \widehat{\boldsymbol{x}} = \boldsymbol{x} + \triangle, \quad ||\widehat{\boldsymbol{x}}||_2 = 1,$$

in the least squares sense.

Gauss-Newton:

$$\sum_{j=1}^{n} \widehat{x}_j B_j = \triangle\lambda y^\top + \lambda \triangle y^\top, \quad \widehat{x} = x + \triangle, \quad ||\widehat{x}||_2 = 1.$$

Implementation Idea No. 1: exclude $\triangle y$ and $\triangle \lambda_k$:

$$Hy = he_1, \quad C\lambda = ce_1$$

using Householder matrices H and C (of sizes $n \times n$ and $r \times r$) such that

$$\sum_{j=1}^{n} \widehat{x}_j \widehat{B}_j = ce_1 \triangle \widehat{y}^\top + h \triangle \widehat{\lambda} e_1^\top, \qquad (*)$$

$$\widehat{B}_j = C B_j H^\top, \quad \triangle \widehat{y} = H \triangle y, \quad \triangle \widehat{\lambda} = C \triangle \lambda.$$

Problem $(*)$ is split into two independent problems:

- **To find $\widehat{x}$, minimize $||\sum_{j=1}^{n} B_j x_j||_F^2$, $||x|| = 1$, where the matrices b_j are obtained from $\widehat{a}_j$ by replacing the elements in the first row and column by zeroes.**
- **Then, $\triangle\widehat{y}$ and $\triangle\widehat{\lambda}$ can be determined from the equations**

$$(\sum_{j=1}^{n} \widehat{x}_j \widehat{B}_j)_{k1} = h\triangle\widehat{\lambda}_k,\ k = 2, ..., r, \quad (\sum_{j=1}^{n} \widehat{x}_j \widehat{B}_j)_{1i} = c\triangle\widehat{y}_i,\ i = 2, ..., n.$$

For the two unknowns $\triangle\widehat{y}_1$ and $\triangle\widehat{\lambda}_1$, we have only one equation, so one of these unknowns can be chosen arbitrary.

Implementation Idea No. 2:

Our main problem is one of finding the minimal singular value of a matrix

$$\boldsymbol{B} = [\mathrm{vec}(\boldsymbol{B}_1), ..., \mathrm{vec}(\boldsymbol{B}_n)],$$

where the operator vec transforms a matrix into a vector taking the elements column-by-column.

Therefore, $\widehat{\boldsymbol{x}}$ is an eigenvector (normalized to have a unit norm) for the minimal eigenvalue of the $n \times n$ matrix $\Gamma = B^\top B$:

$$\boldsymbol{\Gamma \widehat{x} = \gamma_{\min} \widehat{x}.}$$

The elements of Γ are given by

$$\boldsymbol{\Gamma_{sl} = (B_s, B_l)_F}$$

where $(\cdot, \cdot)_F$ is the Frobenius (Euclidian) scalar product of matrices. To calculate the new vector $\widehat{\boldsymbol{x}}$, we need to find the minimal eigenvalue and its eigenvector of Γ.

Solution consists of the two parts:

1. Calculation of the matrix $\mathbf{\Gamma}$.
2. Finding the minimal eigenvalue and the corresponding eigenvector of the matrix $\mathbf{\Gamma}$.

Since only one eigenvector for $\boldsymbol{\Gamma}$ is to be found, we propose to use the shifted inverse iteration using $\boldsymbol{x}$ from the previous iteration as an initial guess.

COMPLEXITY = $\boldsymbol{\mathcal{O}(n^3)}$.

Straitforward implementation of Step 1 includes
$\boldsymbol{\mathcal{O}(n^2r + nr^2)}$ (calculation of $\boldsymbol{B_j}$) + $\boldsymbol{\mathcal{O}(n^2rn)}$(calculation of the $\boldsymbol{B^\top B}$) arithmetic operations.

The total cost of the step 1 is

$$\boldsymbol{\mathcal{O}(n^3r + n^2r + nr^2).}$$

IMPORTANT OBSERVATION:
$\boldsymbol{\Gamma}$ can be computed a way more efficiently without the explicit computation of the Householder matrices.

NUMERICAL EXPERIMENTS

Generate random two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ of order $\boldsymbol{n}$ and $\boldsymbol{r}$ diagonal matrices $\boldsymbol{\Lambda_k}$, $\boldsymbol{k = 1, ..., r}$, of the same order, and set

$$\boldsymbol{A_k = X \Lambda_k Y, \quad k = 1, ..., r.}$$

The elements of $\boldsymbol{X}$ $\boldsymbol{Y}$ and $\boldsymbol{\Lambda_k}$ are drawn from the uniform distribution on $\boldsymbol{[-1, 1]}$. De Lathauwer showed that these sequence of matrices have an exact $\boldsymbol{SGSD}$, because we can find orthogonal $\boldsymbol{Q}$ and $\boldsymbol{Z}$ such that

$$\boldsymbol{X = QR_1, Y = R_2 Z,}$$

with $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ being upper triangular.

We also corrupt these matrices with multiplicative noise, setting

$$\boldsymbol{(\widehat{A}_k)_{ij} = (A_k)_{ij}(1 + \sigma\phi),}$$

where $\boldsymbol{\phi}$ are taken from the uniform distribution on $\boldsymbol{[-1, 1]}$ and $\boldsymbol{\sigma}$ is a "noise level".

We are interested in the following quantities:

- The convergence speed, its dependence from n,r, and $\boldsymbol{\sigma}$.
- The stability: the dependence of the *residue* of the SGSD from $\boldsymbol{\sigma}$.

We have observed that the speed of the algorithm does not depend any pronouncedly on $\boldsymbol{\sigma}$. We perform two experiments:

- First, we fix $\boldsymbol{r}$ and $\boldsymbol{\sigma}$ setting them to $\mathbf{10}$ and $\mathbf{10^{-6}}$ respectively and change $\boldsymbol{n}$. The timings (in seconds) are given in Table 1.
- Second, we set r to be equal to n.
  Corresponding timings are given in Table 2.

To check stability we take fixed $\boldsymbol{r = n = 64}$ and vary the noise level. For each noise level 10 test sequences of matrices are generated and the mean, maximal and minimum values of the residue are reported.

Table 1 Timings(in seconds) for the computation of SGCD of 10 n-by-n matrices.

n	Time
16	0.01
32	0.11
64	1.6
128	14.77
256	210.61

Table 2 Timings(in seconds) for the computation of SGCD of n n-by-n matrices.

n	Time
16	0.02
32	0.14
64	3.41
128	54.96
256	810.77

Table 3 Residues for different noise levels.

σ	Mean residue	Min residue	Max residue
10^{-16}	$2 \cdot 10^{-15}$	$9 \cdot 10^{-16}$	$5 \cdot 10^{-15}$
10^{-15}	$7 \cdot 10^{-15}$	$1 \cdot 10^{-15}$	$2 \cdot 10^{-14}$
10^{-14}	$3 \cdot 10^{-14}$	$4 \cdot 10^{-15}$	$8 \cdot 10^{-14}$
10^{-13}	$5 \cdot 10^{-14}$	$4 \cdot 10^{-14}$	$8 \cdot 10^{-14}$
10^{-12}	$2 \cdot 10^{-12}$	$4 \cdot 10^{-13}$	$4 \cdot 10^{-12}$
10^{-11}	$6 \cdot 10^{-11}$	$4 \cdot 10^{-12}$	$1 \cdot 10^{-11}$
10^{-10}	$4 \cdot 10^{-10}$	$4 \cdot 10^{-11}$	$1 \cdot 10^{-9}$
10^{-9}	$2 \cdot 10^{-8}$	$4 \cdot 10^{-10}$	$5 \cdot 10^{-8}$
10^{-8}	$4 \cdot 10^{-8}$	$4 \cdot 10^{-9}$	$1 \cdot 10^{-7}$
10^{-7}	$2 \cdot 10^{-7}$	$4 \cdot 10^{-8}$	$7 \cdot 10^{-7}$
10^{-6}	$6 \cdot 10^{-7}$	$4 \cdot 10^{-7}$	$1 \cdot 10^{-6}$
10^{-5}	$2 \cdot 10^{-5}$	$4 \cdot 10^{-6}$	$5 \cdot 10^{-5}$
10^{-4}	$4 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1 \cdot 10^{-3}$
10^{-3}	$2 \cdot 10^{-3}$	$4 \cdot 10^{-4}$	$6 \cdot 10^{-3}$