Workshop on Algorithms for Modern Massive Data Sets

Stanford University and Yahoo! Research, June 21–24, 2006

# TENSOR COMPRESSION FOR PETABYTE-SIZE DATA


Eugene Tyrtyshnikov

(joint work with Ivan Oseledets and Dmitry Savostyanov)

Institute of Numerical Mathematics

Russian Academy of Sciences

## NONLOCAL DEPENDENCIES IN DATA

$$\int_D G(x,y)\phi(y)dy = f(x), \qquad x, y \in D \subset R^d$$

| | | |
|---|---|---|
| $d=1$ | $x=(x_i)$ | $\mathcal{A}=[g(x_i,y_j)]=[a_{ij}]$ |
| $d=2$ | $x=(x_{i_1,i_2})$ | $\mathcal{A}=[g(x_{i_1i_2},y_{j_1j_2})]=[a_{i_1i_2j_1j_2}]=[a_{(i_1j_1)(i_2j_2)}]$ |
| $d=2$ | $x=(x_{i_1,i_2,i_3})$ | $\mathcal{A}=[g(x_{i_1i_2i_3},y_{j_1j_2j_3})]=[a_{i_1i_2j_1j_2i_3j_3}]=[a_{(i_1j_1)(i_2j_2)(i_3j_3)}]$ |

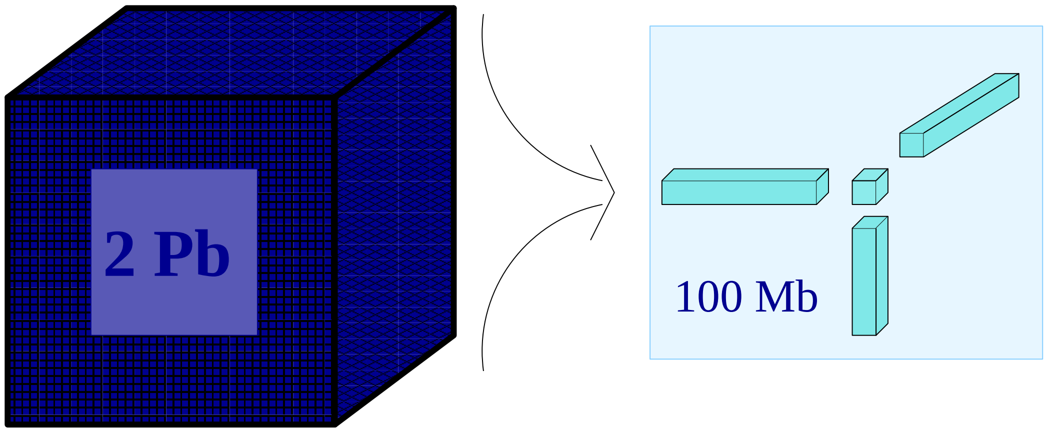
| | | |
|---|---|---|
| $d=1$ | $\mathcal{A}=[a_{ij}]$ | matrix |
| $d=2$ | $\mathcal{A}=[a_{(i_1j_1)(i_2j_2)}]$ | $4$–tensor / matrix |
| $d=2$ | $\mathcal{A}=[a_{(i_1j_1)(i_2j_2)(i_3j_3)}]$ | $6$–tensor / $3$–tensor |

## LARGE-SCALE ARRAYS

| DIMENSION | GRID SIZE | MATRIX SIZE | $n$ for $\texttt{mem}=1$Gb | $\texttt{mem}$ for $n=128$ |
|---|---|---|---|---|
| 2D BEM, $d=1$ | $n$ | $\texttt{mem}=n^2$ | $11000$ | $125$ Kb |
| 3D BEM, $d=2$ | $n^2$ | $\texttt{mem}=n^4$ | $100$ | $2$ Gb |
| 3D VEM, $d=3$ | $n^3$ | $\texttt{mem}=n^6$ | $23$ | $32$ Tb |

## IDEA FOR TENSOR COMPRESSION

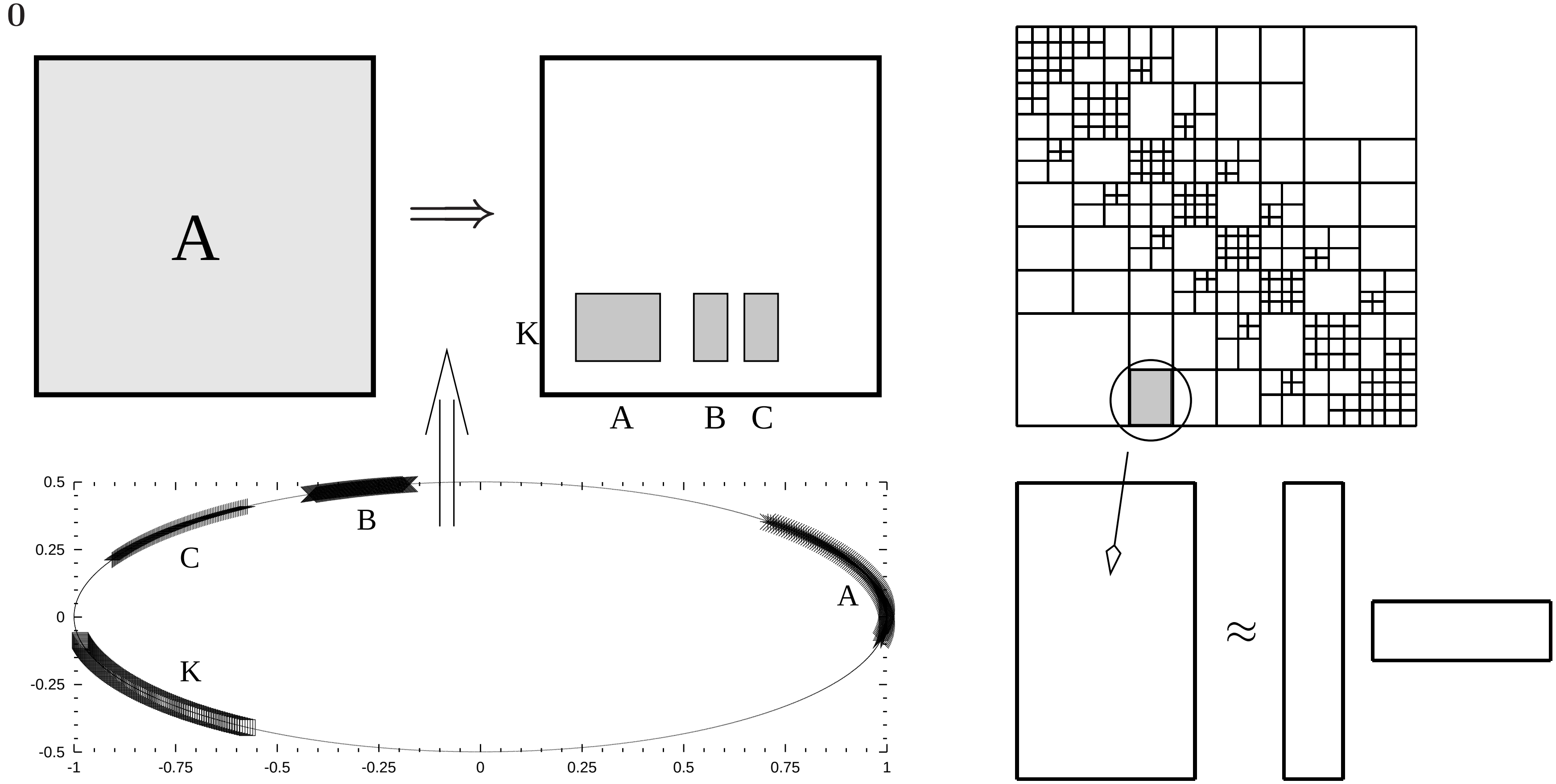
### SEPARATE VARIABLES!



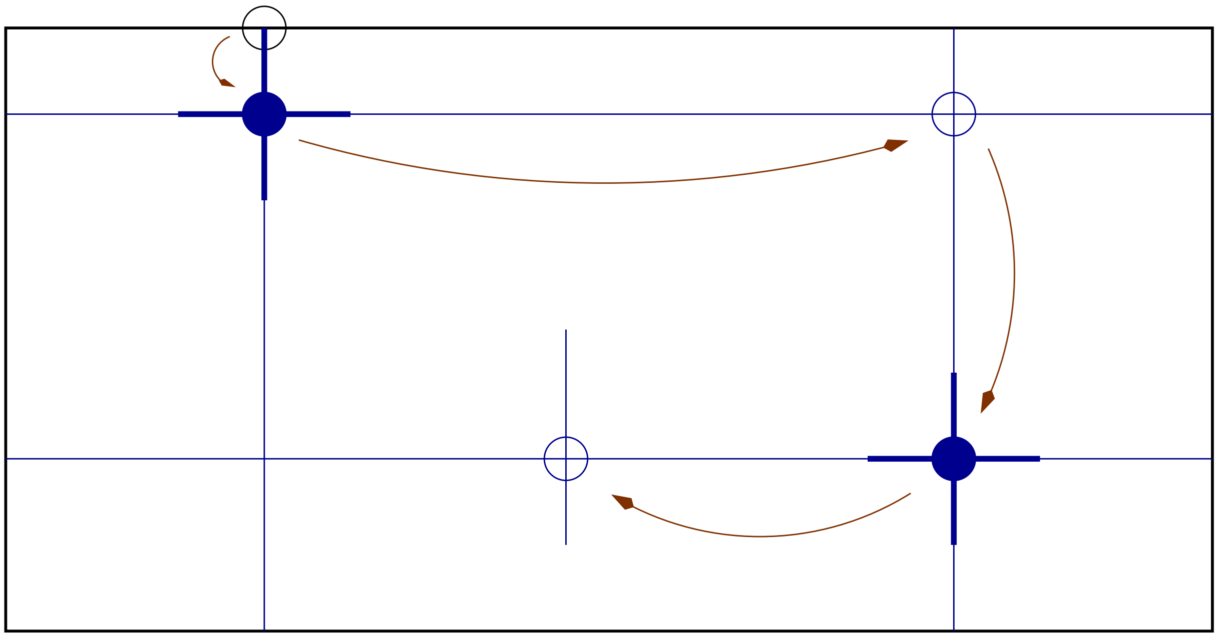

### USE ONLY SMALL PORTION OF DATA!

# RANK STRUCTURED APPROXIMATIONS

$$\int_0^{2\pi} \Phi(x,y)\phi(y)ds_y = f(x), \quad x, y \in \partial\Omega \quad \Longrightarrow \quad Ax = b$$



# 2D CROSS APPROXIMATION



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

**0** Initialization: $\boldsymbol{p} = \mathbf{1},\ \boldsymbol{j}_1 = \mathbf{1}.$

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

**2** Compute row $\boldsymbol{i}_{\boldsymbol{p}}$, subtract current approximation values $\tilde{\boldsymbol{A}}_{\boldsymbol{p}}$. Find pivot $\boldsymbol{j}_{\boldsymbol{p}+1} \neq \boldsymbol{j}_{\boldsymbol{p}}$.

**3** Using the cross $(\boldsymbol{i}_{\boldsymbol{p}}, \boldsymbol{j}_{\boldsymbol{p}})$, construct a new skeleton annihilating this cross.

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

# 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

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


- 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.

# APPROXIMATION OF MATRICES AND 3D ARRAYS

Reshaping (reordering of multi-indices):

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

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

Tensor approximation of a matrix:

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

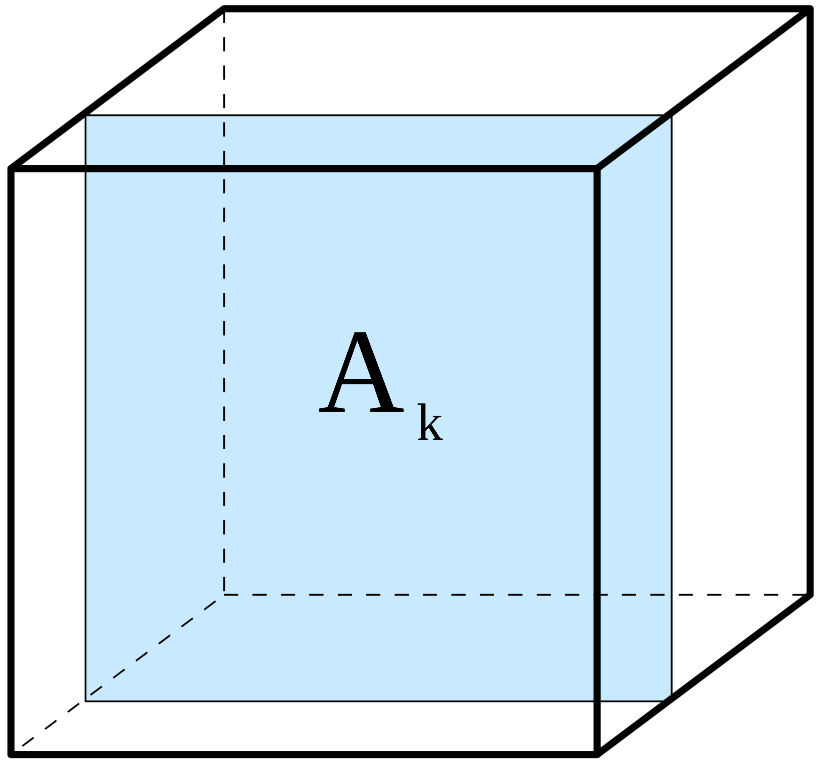
Trilinear approximation of a tensor (3D array):

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

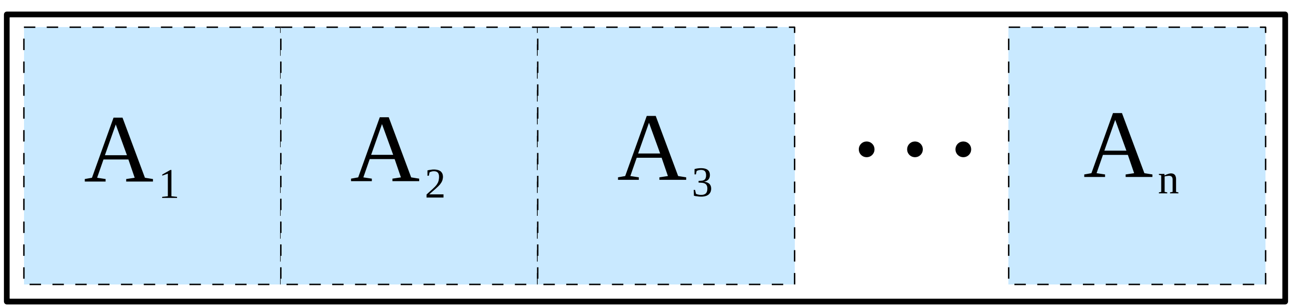
**Tensor approx. of a matrix $\Leftrightarrow$ Trilinear approx. of a 3D array**

# 3D ARRAYS AND MATRICES

## Slices in one mode




## Matrices of slices



## TRILINEAR DECOMPOSITION

$$\boldsymbol{a_{ijk} = \sum_{t=1}^{r} u_{it} v_{jt} w_{kt}.}$$

## MINIMIZATION ALGORITHMS

$$\boldsymbol{\min_{u,v,w} \sum_{i,j,k} \left( \sum_{t=1}^{r} u_{it} v_{jt} w_{kt} - a_{ijk} \right)^2 .}$$

[+] **standard algorihms**: ALS, Gauß-Newton, Damped LM Newton.

[−] **costly** iteration, **slow** convergence without a good initial guess.

[−] necessity of **a priori** knowledge of $\boldsymbol{r}$.

## TRILINEAR DECOMPOSITION

$$a_{ijk} = \sum_{t=1}^{r} u_{it} v_{jt} w_{kt}.$$

## MATRIX-BASED ALGORITHMS

- $r = n$

Generalized Schur decompostion

$$\mathcal{A} = [A_k], \qquad A_k = UB_kV^\top, \quad Q^\top A_k Z = RB_kL^\top.$$

[+] Fast algorithms fetching a good initial guess

[−] Restriction: $r = n$.

- $r > n$

*Matrix methods for overdetermined cases*
are not discussed in the literature.

# TRILINEAR DECOMPOSITION

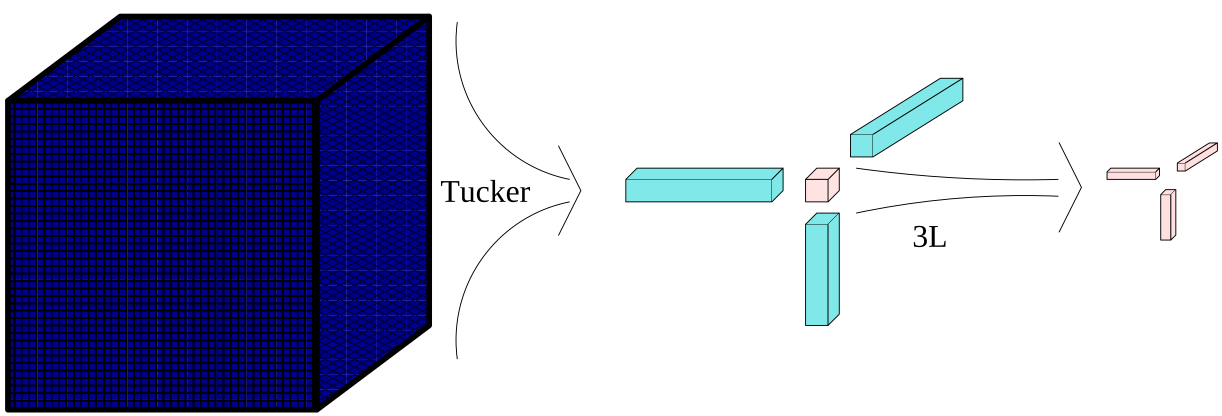
All methods are too slow
for $\boldsymbol{n \geq 128}$.

## 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

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

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

# 3D CROSS EXISTENCE THEOREM

Suppose we are aware that

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

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

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

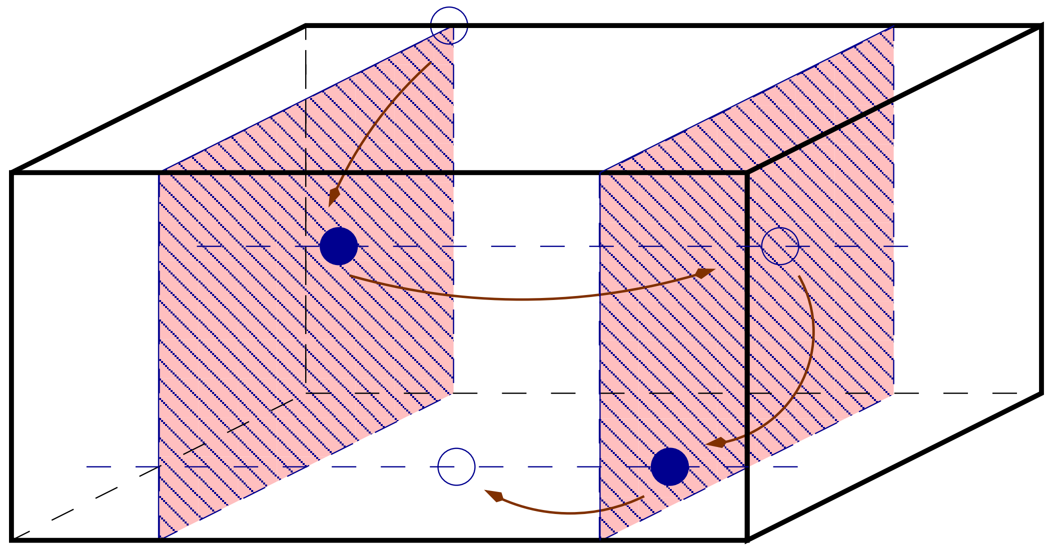
where

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

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 $\boldsymbol{\mathcal{A}} \approx \tilde{\boldsymbol{\mathcal{A}}} = \sum_{q=1}^{r} \boldsymbol{A}_q \times \boldsymbol{w}_q$.

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

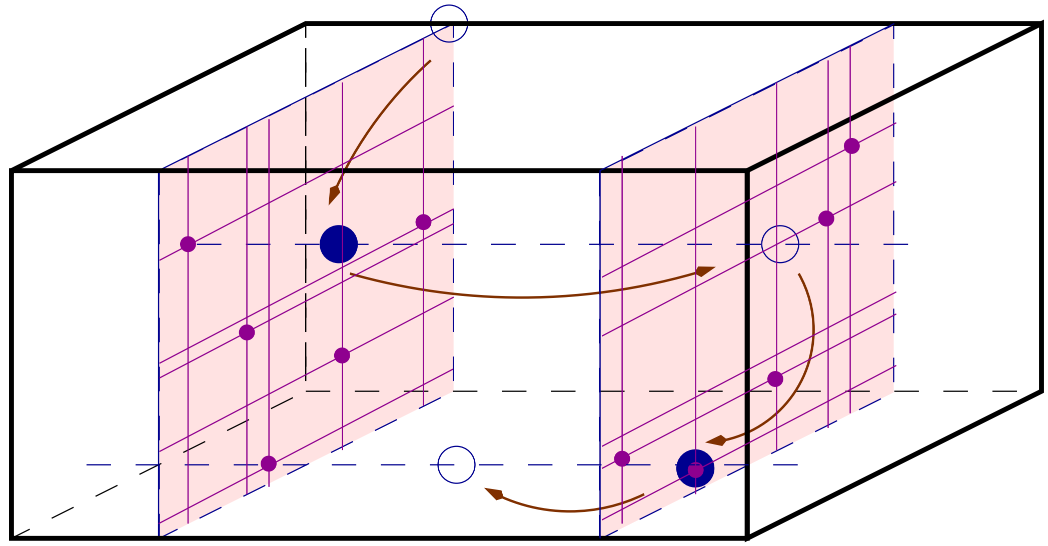
**2** Compute a fiber $\boldsymbol{w}_p$, subtract approx. values $\tilde{\boldsymbol{\mathcal{A}}}$. Find pivot $\boldsymbol{k}_{p+1} \neq \boldsymbol{k}_p$.

**3** Skeleton $\boldsymbol{A}_{k_p} \times \boldsymbol{w}_p$ nullifies the slice-by-fiber cross.

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

# 3D CROSS APPROXIMATION

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



Find $\boldsymbol{\mathcal{A}} \approx \tilde{\boldsymbol{\mathcal{A}}} = \sum_{q=1}^{r^2} \boldsymbol{u}_q \times \boldsymbol{v}_q \times \boldsymbol{w}_q.$

**1** Compute a *cross approximation* of the slice $\boldsymbol{A}_{k_p} = \sum_{q=1}^{r} \boldsymbol{u}_{pq}\boldsymbol{v}_{pq}^\top,$
subtract approx. values $\tilde{\boldsymbol{\mathcal{A}}}.$
Find pivot $\boldsymbol{A}_{k_p}$ (HOW CAN WE DO THIS?)

**2** Compute a fiber $\boldsymbol{w}_p,$ subtract approx. values $\tilde{\boldsymbol{\mathcal{A}}}.$ Find pivot $\boldsymbol{k}_{p+1} \neq \boldsymbol{k}_p.$

**3** Skeleton $\sum^{r} \boldsymbol{u}_{pq} \times \boldsymbol{v}_{pq} \times \boldsymbol{w}_p$ nullifies the slice-by-fiber cross.

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

| | | $1 \cdot 10^{-3}$ | | $1 \cdot 10^{-5}$ | | $1 \cdot 10^{-7}$ | | $1 \cdot 10^{-9}$ | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | 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

| | | 1.10−3 | | 1.10−5 | | 1.10−7 | | 1.10−9 | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | 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

Asymptotically smooth generating function: $\boldsymbol{a_{ij} = F(\mathbf{src}_i - \mathbf{obs}_j)}$

$$|D^{\mathrm{p}} F(v)| \le c d^p p! \|v\|^{g-p}, \qquad \forall p \ge 0.$$

$$\mathrm{p} = (p_1, \ldots, p_m), \quad p = p_1 + \ldots + p_m, \qquad D^{\mathrm{p}} = \frac{\partial^{p_1} \ldots \partial^{p_m}}{(\partial v_1)^{p_1} \ldots (\partial v_m)^{p_m}}.$$

Tensor grids:

$$\mathbf{src}_i = (x_1, \ldots, x_d), \quad \mathbf{obs}_i = (y_1, \ldots, y_d)$$

## THEOREM.

$$r \le \left(c_0 + c_1 \log h^{-1}\right) p^{d-1} + \tau,$$
$$|\{A - \tilde{A}_r\}_{ij}| \le c_2 \gamma^p \|\mathbf{src}_i - \mathbf{obs}_j\|^g.$$

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).

## THEORY: TENSOR RANK ESTIMATES

Approximation error ($\boldsymbol{d = 3}$)

$$\boldsymbol{\varepsilon_{\mathrm{abs}} = c_2 \gamma^p \|v\|^g, \qquad \varepsilon = \varepsilon_{\mathrm{rel}} = c_2 \gamma^p}$$

$$\boldsymbol{r \leq \left(c_0 + c_1 \log h^{-1}\right)\left(c_3 \log \varepsilon^{-1} + c_4\right)^2 + \tau.}$$

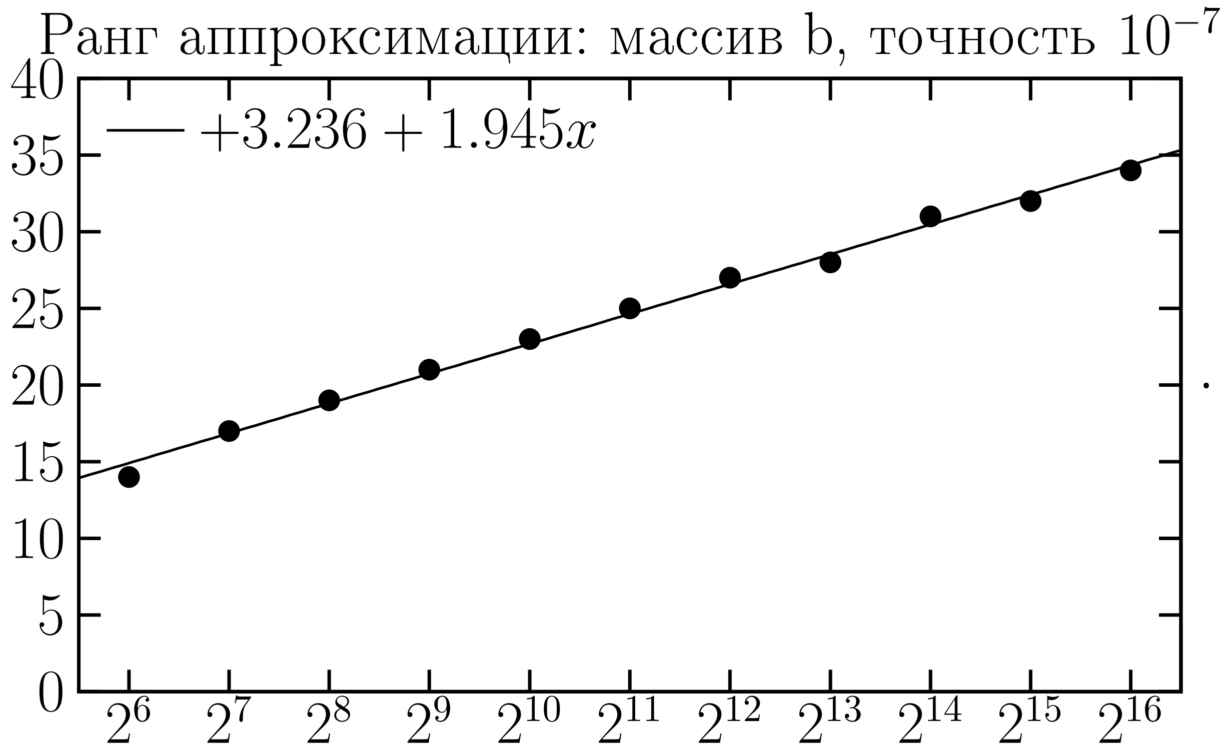
On *almost uniform* grids $\boldsymbol{h^{-1} \sim n}$

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

# 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

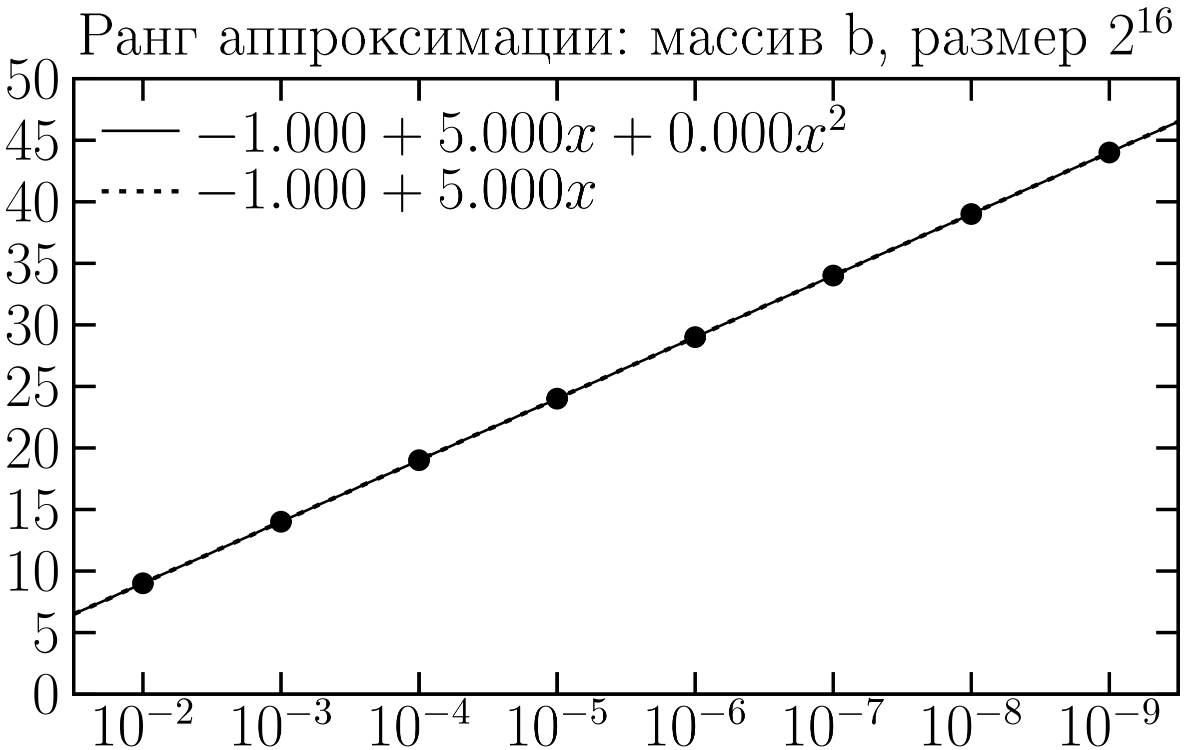



$$r \sim \log n$$

# PRACTICAL PROOF

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

## Tensor rank versus approximation error



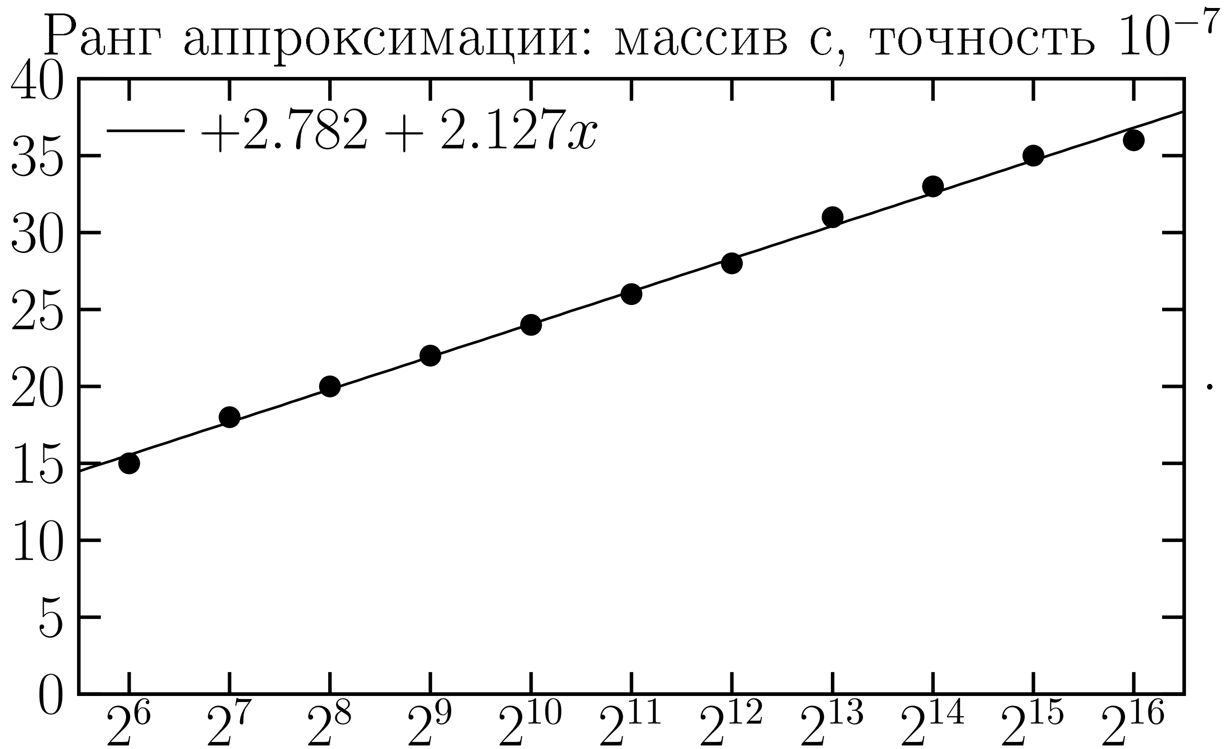

.

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

# PRACTICAL PROOF

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

## Tensor rank versus array size

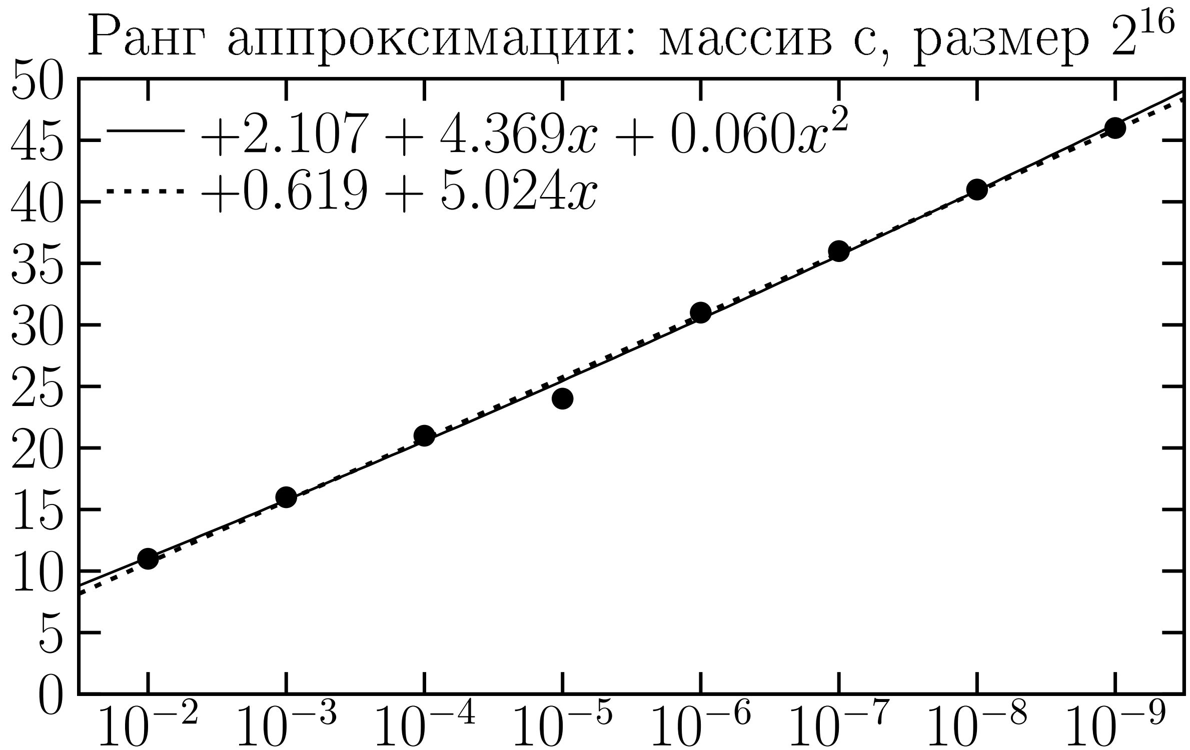



$$r \sim \log n$$

# PRACTICAL PROOF

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

## Tensor rank versus approximation error




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

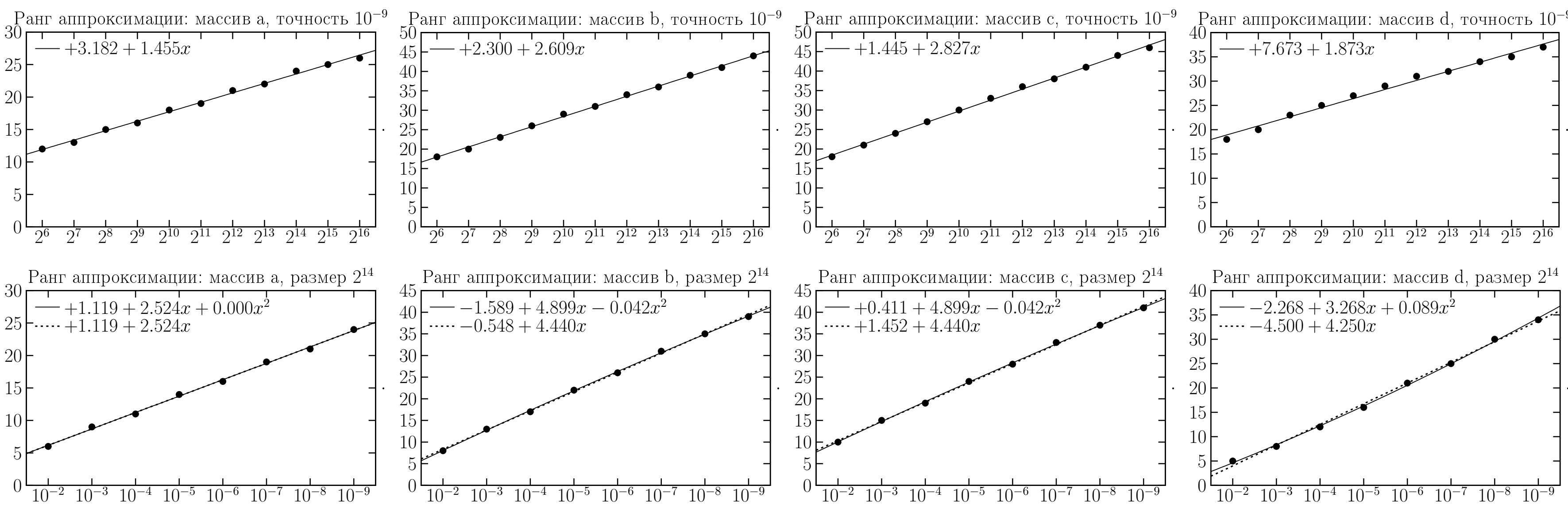
# ASYMPTOTICS OF TENSOR RANK

## Theory

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

## Practice

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



# TENSOR SOLVER WITH TENSOR VECTORS

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

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

| grid size $n$ | **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 |

## FAST SIMULTANEOUS ORTHOGONAL REDUCTION TO TRIANGULAR MATRICES

*Given* $\boldsymbol{n \times n}$ *real matrices* $\boldsymbol{A_1, \ldots, A_r}$, *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* $\boldsymbol{Q}\boldsymbol{A}_1\boldsymbol{Z}$, ..., $\boldsymbol{Q}\boldsymbol{A}_r\boldsymbol{Z}$ *as upper triangular as possible.*

### DEFLATION STEP:

$$\boldsymbol{Q}\boldsymbol{A}_k\boldsymbol{Z} \approx \begin{pmatrix} \boldsymbol{\lambda}_k & \boldsymbol{v}_k^\top \\ \mathbf{0} & \boldsymbol{B}_k \end{pmatrix} \qquad \Leftrightarrow \qquad \boldsymbol{Q}\boldsymbol{A}_k\boldsymbol{Z}\boldsymbol{e}_1 \approx \boldsymbol{\lambda}_k\boldsymbol{e}_1$$

$$\boxed{\boldsymbol{A}_k\boldsymbol{x} = \boldsymbol{\lambda}_k\boldsymbol{y}, \qquad \boldsymbol{x} = \boldsymbol{Z}\boldsymbol{e}_1, \quad \boldsymbol{y} = \boldsymbol{Q}^\top\boldsymbol{e}_1.}$$

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

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

## Gauss-Newton algorithm for the simultaneous eigenvalue problem

$$\sum_{j=1}^{n} A_{ij}^{k} x_j = \lambda_k y_i. \tag{1}$$

Introduce $r \times n$ matrices

$$(a_j)_{ki} = A_{ij}^{k}, k = 1, ..., r, \ , i = 1, ..., n \ , j = 1, ..., n,$$

and a column vector $\lambda = [\lambda_1, \ ..., \ \lambda_r]^\top$. Then (1) becomes

$$\sum_{j=1}^{n} x_j a_j = \lambda y^\top. \tag{2}$$

**Gauss-Newton:** linearize the system producing an overdetermined linear system and then solve it in the least squares sense.

$$\sum_{j=1}^{n} \widehat{x}_j a_j = \triangle \lambda y^\top + \lambda \triangle y^\top, \quad \widehat{x} = x + \triangle, \quad ||\widehat{x}||_2 = 1. \tag{3}$$

**Gauss-Newton:**

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

**Exclude $\triangle \boldsymbol{y}$ and $\triangle \boldsymbol{\lambda}_k$:**

$$\boldsymbol{H}\boldsymbol{y} = \boldsymbol{h}\boldsymbol{e}_1, \quad \boldsymbol{C}\boldsymbol{\lambda} = \boldsymbol{c}\boldsymbol{e}_1$$

using Housholder matrices $\boldsymbol{H}$ and $\boldsymbol{C}$ (of sizes $\boldsymbol{n} \times \boldsymbol{n}$ and $\boldsymbol{r} \times \boldsymbol{r}$) such that

$$\sum_{j=1}^{n} \widehat{\boldsymbol{x}}_j \widehat{\boldsymbol{a}}_j = \boldsymbol{c}\boldsymbol{e}_1 \triangle \widehat{\boldsymbol{y}}^\top + \boldsymbol{h} \triangle \widehat{\boldsymbol{\lambda}} \boldsymbol{e}_1^\top, \tag{4}$$

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

Problem (4) is split into two independent problems:

- **To find $\widehat{x}$, minimize $|| \sum_{j=1}^{n} \boldsymbol{b}_j \boldsymbol{x}_j ||_F^2, \quad ||x|| = 1$, where the matrices $\boldsymbol{b}_j$ are obtained from $\widehat{\boldsymbol{a}}_j$ by replacing the elements in the first row and column by zeroes.**
- **Then, $\triangle \widehat{\boldsymbol{y}}$ and $\triangle \widehat{\boldsymbol{\lambda}}$ can be determined from the equations**
  $$(\sum_{j=1}^{n} \widehat{\boldsymbol{x}}_j \widehat{\boldsymbol{a}}_j)_{k1} = \boldsymbol{h} \triangle \widehat{\boldsymbol{\lambda}}_k, \ \boldsymbol{k} = 2, ..., \boldsymbol{r}, \quad (\sum_{j=1}^{n} \widehat{\boldsymbol{x}}_j \widehat{\boldsymbol{a}}_j)_{1i} = \boldsymbol{c} \triangle \widehat{\boldsymbol{y}}_i, \ \boldsymbol{i} = 2, ..., \boldsymbol{n}.$$

For the two unknowns $\triangle\widehat{\boldsymbol{y}}_{\mathbf{1}}$ and $\triangle\widehat{\boldsymbol{\lambda}}_{\mathbf{1}}$, we have only one equation, so one of these unknowns can be chosen arbitrary.

Having obtained the new $\widehat{\boldsymbol{x}}$, we propose to evaluate $\boldsymbol{y}$ and $\boldsymbol{\lambda}$ by the power method as follows:

$$\widetilde{\boldsymbol{\lambda}} = \boldsymbol{b}\boldsymbol{y}, \quad \widetilde{\boldsymbol{y}} = \boldsymbol{b}^\top\boldsymbol{\lambda}, \tag{5}$$

where

$$\boldsymbol{b} = \sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}} \widehat{\boldsymbol{x}}_j \boldsymbol{a}_j.$$

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

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

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 $\boldsymbol{n} \times \boldsymbol{n}$ matrix $\boldsymbol{\Gamma} = \boldsymbol{B}^\top \boldsymbol{B}$:**

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

**The elements of $\boldsymbol{\Gamma}$ are given by**

$$\boldsymbol{\Gamma}_{sl} = (\boldsymbol{b}_s, \boldsymbol{b}_l)_F$$

**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 $\boldsymbol{\Gamma}$.**

Solution consists of the two parts:

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

Since only one eigenvector for $\mathbf{\Gamma}$ 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 $\boldsymbol{\mathcal{O}(n^2 r + n r^2)}$ (calculation of $\boldsymbol{b_j}$) + $\boldsymbol{\mathcal{O}(n^2 r n)}$(calculation of the $\boldsymbol{B}^\top \boldsymbol{B}$) arithmetic operations.

The total cost of the step 1 is

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

However, $\mathbf{\Gamma}$ can be computed a way more efficiently without the explicit computation of the Householder matrices.