

6th WSC: Modern methods of data analysis

FAST COMPUTATION OF CANDECOMP-PARAFAC
AND TUCKER DECOMPOSITIONS

Dmitry Savostyanov, Institute of Numerical Mathematics of RAS, Moscow
joint work with S. Goreinov, I. Oseledets and E. Tyrtishnikov.

Russia, Kazan, 18-22/02/2008

CANDECOMP-PARAFAC

Simple form for 2D arrays (matrices), known as *skeleton* or ‘low-rank’ decomposition:

$$a_{ij} = \sum_{\alpha=1}^r u_{i\alpha} v_{j\alpha} \Leftrightarrow A = UV^T, \quad r = \text{rank}(A)$$

General form of canonical decomposition of data array, also known as ‘multilinear’ decomposition:

$$a_{ij\dots k} = \sum_{\alpha=1}^r u_{i\alpha} v_{j\alpha} \dots w_{k\alpha}, \quad r = \text{Rank}(\mathcal{A})$$

WHY Rank(TENSOR) IS NOT AS SIMPLE AS rank(MATRIX)?

- $\text{Rank}(\mathcal{A})$ in \mathbb{R} may differ from $\text{Rank}(\mathcal{A})$ in \mathbb{C} for the same \mathcal{A} .
- We don’t know finite algorithm to compute $\text{Rank}(\mathcal{A})$.
- We don’t know exact upper bound for $\text{Rank}(\mathcal{A})$.
- 100% of 2×2 matrices have rank 2, but
79% of $2 \times 2 \times 2$ tensors have Rank 2 and 21% have Rank 3.

WHY Rank IS IMPORTANT?

It shows you the number of components in the mixture (ICA method), shows the number of main features in the ensemble (of experiments, of photos, of i-don’t-know what else), give key to effective data storage in all these cases.

TUCKER DECOMPOSITION

Also known as HO-SVD decomposition.

$$\mathbf{a}_{ij\dots k} = \sum_{i'=1}^{r_i} \sum_{j'=1}^{r_j} \dots \sum_{k'=1}^{r_k} g_{i'j'\dots k'} \mathbf{u}_{ii'} \mathbf{v}_{jj'} \dots \mathbf{w}_{kk'} \quad \Leftrightarrow \quad \mathcal{A} = \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \dots \times_d \mathbf{W}.$$

Simple form for matrices, known as Shur or SVD decomposition:

$$\mathbf{a}_{ij} = \sum_{i'=1}^r \sum_{j'=1}^r g_{i'j'} \mathbf{u}_{ii'} \mathbf{v}_{jj'} = (\text{SVD makes } G \text{ diagonal}) = \sum_{\alpha=1}^r \mathbf{u}_{i\alpha} \sigma_{\alpha} \mathbf{v}_{j\alpha}.$$

WHY TUCKER DECOMPOSITION IS NOT AS DIFFICULT AS CANONICAL?

- Mode ranks r_i, r_j, \dots, r_k can be computed via SVD.
- Mode factors $\mathbf{U}, \mathbf{V} \dots \mathbf{W}$ can be computed via SVD

Mode factors can be chosen unitary \Rightarrow 'uniqueness' of decomposition.

WHY TUCKER DECOMPOSITION IS IMPORTANT?

It is: more compact, more stable, know-how to compute. When you have Tucker, it is easy to filter non-important subspace and find low-rank approximation with guaranteed accuracy. When you have Tucker decomposition with small core \mathcal{G} , it is easier to make CANDECAMP.

$$\begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix} =$$

$$= \begin{pmatrix} -.66 & 0.67 & 0.31 & -.09 & 0.01 \\ -.48 & -.81 & -.68 & 0.51 & -.16 \\ -.38 & -.34 & -.26 & -.58 & 0.56 \\ -.32 & -.44 & 0.20 & -.33 & -.73 \\ -.27 & -.47 & 0.56 & 0.51 & 0.32 \end{pmatrix} \cdot \begin{pmatrix} 1.056 \cdot 10^0 \\ 0.82 \cdot 10^{-1} \\ 0.33 \cdot 10^{-2} \\ 0.74 \cdot 10^{-4} \\ 0.69 \cdot 10^{-6} \end{pmatrix} \cdot \begin{pmatrix} -.66 & -.48 & -.38 & -.32 & -.27 \\ 0.67 & -.08 & -.34 & -.44 & -.47 \\ 0.31 & -.68 & -.26 & 0.20 & 0.56 \\ -.08 & .51 & -.58 & -.33 & 0.51 \\ 0.01 & -.16 & 0.56 & -.73 & 0.32 \end{pmatrix}$$

$$\begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix} =
\begin{pmatrix} -.66 & 0.67 & 0.31 & -.09 & 0.01 \\ -.48 & -.81 & -.68 & 0.51 & -.16 \\ -.38 & -.34 & -.26 & -.58 & 0.56 \\ -.32 & -.44 & 0.20 & -.33 & -.73 \\ -.27 & -.47 & 0.56 & 0.51 & 0.32 \end{pmatrix} \cdot
\begin{pmatrix} 1.056 \cdot 10^0 & & & & \\ & 0.82 \cdot 10^{-1} & & & \\ & & 0.33 \cdot 10^{-2} & & \\ & & & 0.74 \cdot 10^{-4} & \\ & & & & 0.69 \cdot 10^{-6} \end{pmatrix} \cdot
\begin{pmatrix} -.66 & -.48 & -.38 & -.32 & -.27 \\ 0.67 & -.08 & -.34 & -.44 & -.47 \\ 0.31 & -.68 & -.26 & 0.20 & 0.56 \\ -.08 & .51 & -.58 & -.33 & 0.51 \\ 0.01 & -.16 & 0.56 & -.73 & 0.32 \end{pmatrix} \approx$$

$$\approx
\begin{pmatrix} -.66 & 0.67 \\ -.48 & -.81 \\ -.38 & -.34 \\ -.32 & -.44 \\ -.27 & -.47 \end{pmatrix} \cdot
\begin{pmatrix} 1.056 \cdot 10^0 \\ 0.82 \cdot 10^{-1} \end{pmatrix} \cdot
\begin{pmatrix} -.66 & -.48 & -.38 & -.32 & -.27 \\ 0.67 & -.08 & -.34 & -.44 & -.47 \end{pmatrix}$$

BASIC FACTS ABOUT LOW-RANK MATRICES

Consider « $r + \varepsilon$ matrix» $A = UV^T + E$, $\text{rank}(UV^T) = r$, $\|E\|_C \leq \varepsilon$.

MAIN QUESTION

Approximation requires $2nr$ parameters. Can you compute it in $\mathcal{O}(nr^\alpha)$ flops?

BASIC FACTS ABOUT LOW-RANK MATRICES

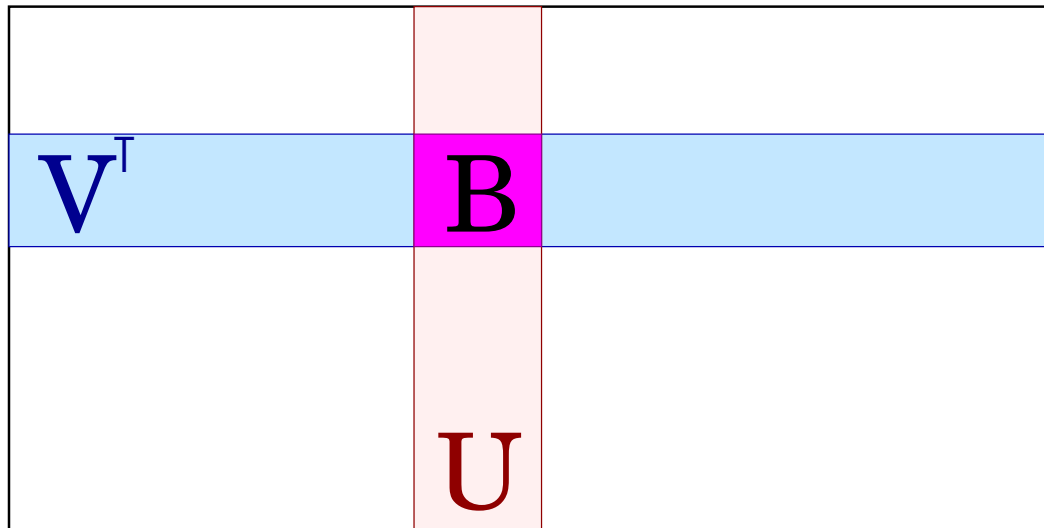
Consider « $r + \varepsilon$ matrix» $A = UV^T + E$, $\text{rank}(UV^T) = r$, $\|E\|_C \leq \varepsilon$.

MAIN QUESTION

Approximation requires $2nr$ parameters. Can you compute it in $\mathcal{O}(nr^\alpha)$ flops?

YES if $E = 0$ (exactly low-rank data):

Take **any** $r \times r$ submatrix B with $\det B \neq 0$ and compute cross approximation



$$A = UB^{-1}V^T \quad \text{exactly.}$$

BASIC FACTS ABOUT LOW-RANK MATRICES

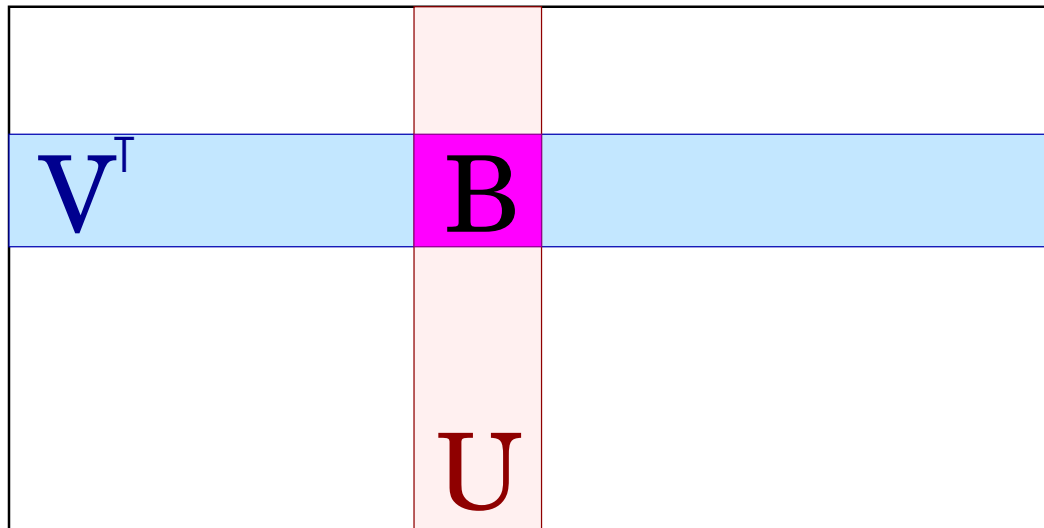
Consider « $r + \varepsilon$ matrix» $A = UV^\top + E$, $\text{rank}(UV^\top) = r$, $\|E\|_c \leq \varepsilon$.

MAIN QUESTION

Approximation requires $2nr$ parameters. Can you compute it in $\mathcal{O}(nr^\alpha)$ flops?

YES for $E \neq 0$ (Goreinov, Tyrtyshnikov, 2001):

If A admits r -rank ε -approximation UV^\top , then it admits also r -rank $(r + 1)\varepsilon$ -approximation with $U = \text{columns}(A)$, $V = \text{rows}(A)$ (**cross approximation**).



$$A = UB^{-1}V^\top + E, \quad \|E\|_c \leq (r + 1)\varepsilon.$$

Here B is **maximum-volume** $r \times r$ submatrix of A . $B = A_{\square}$.

EXAMPLE

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix}$$

MAXIMUM-VOLUME CROSS ($[1, 4] \times [1, 4]$)

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix}$$

MAXIMUM-VOLUME CROSS ($[1, 4] \times [1, 4]$)

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix}$$

CROSS APPROXIMATION

$$A = \begin{pmatrix} 1/2 & 1/5 \\ 1/3 & 1/6 \\ 1/4 & 1/7 \\ 1/5 & 1/8 \\ 1/6 & 1/9 \end{pmatrix} \cdot \begin{pmatrix} 1/2 & 1/5 \\ 1/5 & 1/8 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0030864 & 0.00158730 & 0 & -0.00117578 \\ 0 & 0.0015873 & 0.00085034 & 0 & -0.00066138 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -0.0011758 & -0.000661376 & 0 & 0.0005487 \end{pmatrix}$$

MAXIMUM-VOLUME CROSS ($[1, 4] \times [1, 4]$)

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{pmatrix}$$

CROSS APPROXIMATION

$$A = \begin{pmatrix} 1/2 & 1/5 \\ 1/3 & 1/6 \\ 1/4 & 1/7 \\ 1/5 & 1/8 \\ 1/6 & 1/9 \end{pmatrix} \cdot \begin{pmatrix} 1/2 & 1/5 \\ 1/5 & 1/8 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0030864 & 0.00158730 & 0 & -0.00117578 \\ 0 & 0.0015873 & 0.00085034 & 0 & -0.00066138 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -0.0011758 & -0.000661376 & 0 & 0.0005487 \end{pmatrix}$$

APPROXIMATION ERRORS (Chebyshev norm)

- SVD: 0.0015754;
- Maxvol cross: 0.0030864;
- Best cross approximation: 0.00277777 (for $[1, 3] \times [1, 3]$ cross that is not maxvol).

FAST ALGORITHM FOR CROSS APPROXIMATION should NOT:

- use *exhaustive* search for maximum-volume $r \times r$ submatrix (NP-difficult);
- even *compute* all elements of A (n^2 operations are too much).

FAST ALGORITHM FOR CROSS APPROXIMATION should NOT:

- use *exhaustive* search for maximum-volume $r \times r$ submatrix (NP-difficult);
- even *compute* all elements of A (n^2 operations are too much).

Any ideas?

ADAPTIVE SEARCH FOR THE GOOD CROSS (Tyrttyshnikov, 2000).

1. Choose some column of A and compute it

$$A = \begin{pmatrix} 1/2 & \bullet & \bullet & \bullet & \bullet \\ 1/3 & \bullet & \bullet & \bullet & \bullet \\ 1/4 & \bullet & \bullet & \bullet & \bullet \\ 1/5 & \bullet & \bullet & \bullet & \bullet \\ 1/6 & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

2. Find maximum element in current column

$$A = \begin{pmatrix} 1/2 & \bullet & \bullet & \bullet & \bullet \\ 1/3 & \bullet & \bullet & \bullet & \bullet \\ 1/4 & \bullet & \bullet & \bullet & \bullet \\ 1/5 & \bullet & \bullet & \bullet & \bullet \\ 1/6 & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

3. Compute the corresponding row

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & \bullet & \bullet & \bullet & \bullet \\ 1/4 & \bullet & \bullet & \bullet & \bullet \\ 1/5 & \bullet & \bullet & \bullet & \bullet \\ 1/6 & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

3. Compute the corresponding row

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & \bullet & \bullet & \bullet & \bullet \\ 1/4 & \bullet & \bullet & \bullet & \bullet \\ 1/5 & \bullet & \bullet & \bullet & \bullet \\ 1/6 & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

3'. One-rank approximation is ready:

$$\tilde{A}_1 = \begin{pmatrix} 1/2 \\ 1/3 \\ 1/4 \\ 1/5 \\ 1/6 \end{pmatrix} \cdot (1/2)^{-1} \cdot (1/2 \ 1/3 \ 1/4 \ 1/5 \ 1/6)$$

$$A - \tilde{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

4. Choose next column and compute it

$$A = \begin{pmatrix} 1/2 & \mathbf{1/3} & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & \bullet & \bullet & \bullet \\ 1/4 & 1/5 & \bullet & \bullet & \bullet \\ 1/5 & 1/6 & \bullet & \bullet & \bullet \\ 1/6 & 1/7 & \bullet & \bullet & \bullet \end{pmatrix}$$

4'. Compute residual, find maximum element ...

$$A - \tilde{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1/36 & \bullet & \bullet & \bullet \\ 0 & \mathbf{1/30} & \bullet & \bullet & \bullet \\ 0 & 1/30 & \bullet & \bullet & \bullet \\ 0 & 2/63 & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

5. ...and continue

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & \bullet & \bullet & \bullet \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & \bullet & \bullet & \bullet \\ 1/6 & 1/7 & \bullet & \bullet & \bullet \end{pmatrix}$$

ADAPTIVE SEARCH FOR THE GOOD CROSS

5. ...and continue

$$A = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & \bullet & \bullet & \bullet \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & \bullet & \bullet & \bullet \\ 1/6 & 1/7 & \bullet & \bullet & \bullet \end{pmatrix}$$

WE FINISH, computed $[1, 2] \times [1, 3]$ cross *only!*

COMPARE ERRORS

	det	error
SVD		0.0015754
best cross $[1, 3] \times [1, 3]$	0.020833	0.0027778
maxvol cross $[1, 4] \times [1, 4]$	0.0225	0.0030864
adaptive cross $[1, 2] \times [1, 3]$	0.0166666	0.0047619
random cross	0.0008476	0.04
worst cross $[4, 5] \times [4, 5]$	0.00015432	0.08

COMPARE ERRORS

	det	error
SVD		0.0015754
best cross $[1, 3] \times [1, 3]$	0.020833	0.0027778
maxvol cross $[1, 4] \times [1, 4]$	0.0225	0.0030864
adaptive cross $[1, 2] \times [1, 3]$	0.0166666	0.0047619
random cross	0.0008476	0.04
worst cross $[4, 5] \times [4, 5]$	0.00015432	0.08

IT COSTS nr elements of A and nr^2 flops.

COMPARE ERRORS

	det	error
SVD		0.0015754
best cross $[1, 3] \times [1, 3]$	0.020833	0.0027778
maxvol cross $[1, 4] \times [1, 4]$	0.0225	0.0030864
adaptive cross $[1, 2] \times [1, 3]$	0.0166666	0.0047619
random cross	0.0008476	0.04
worst cross $[4, 5] \times [4, 5]$	0.00015432	0.08

IT COSTS nr elements of A and nr^2 flops.

Can we adjust this to 3D arrays?

TENSORS FROM PCA/ICA

$$\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t), \quad \text{only } \mathbf{y}(t) \text{ is known.}$$

$$\Phi_{\mathbf{y}} = \mathbb{E}[\mathbf{y}(t)\mathbf{y}^\top(t)] = \mathbf{A}[\mathbf{x}(t)\mathbf{x}^\top(t)]\mathbf{A}^\top = \mathbf{A}\Phi_{\mathbf{x}}\mathbf{A}^\top = \Phi_{\mathbf{x}} \times_1 \mathbf{A} \times_2 \mathbf{A}. \quad (\text{pca})$$

$$[\mathcal{C}_{\mathbf{y}}]_{ijkl} = \text{Cum}(\mathbf{y}_i(t), \mathbf{y}_j(t), \mathbf{y}_k(t), \mathbf{y}_l(t)) = [\mathcal{C}_{\mathbf{x}}] \times_1 \mathbf{A} \times_2 \mathbf{A} \times_3 \mathbf{A} \times_4 \mathbf{A}. \quad (\text{ica})$$

Notation:

$$\mathcal{C} = \mathcal{A} \times_2 \mathbf{B} \quad \Leftrightarrow \quad c_{ijkl} = \sum_{j'=1}^n a_{ij'/kl} b_{j'/j}$$

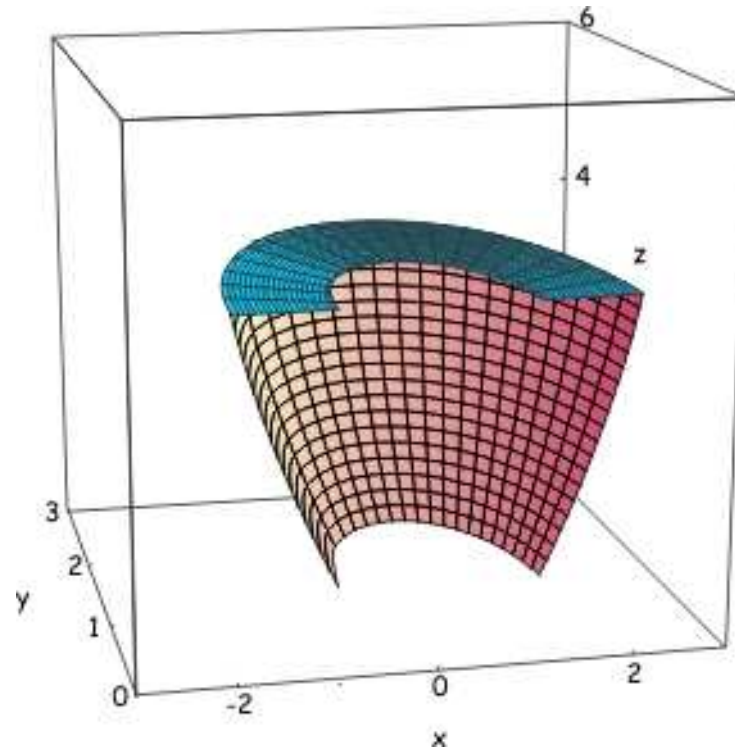
TENSORS FROM INTEGRAL EQUATIONS

Linear system

$$Ax = y, \quad A = [a_{ij}], \quad a_{ij} = f(\text{src}_i, \text{obs}_j),$$

Tensor grid in 3D (volume integral equation):

$$\text{src}_i = (x_{i_1}, y_{i_2}, z_{i_3}), \quad \text{obs}_j = (x_{j_1}, y_{j_2}, z_{j_3}), \quad i = (i_1, i_2, i_3), j = (j_1, j_2, j_3),$$



TENSORS FROM INTEGRAL EQUATIONS

Linear system

$$Ax = y, \quad A = [a_{ij}], \quad a_{ij} = f(\text{src}_i, \text{obs}_j),$$

Tensor grid in 3D (volume integral equation):

$$\text{src}_i = (x_{i_1}, y_{i_2}, z_{i_3}), \quad \text{obs}_j = (x_{j_1}, y_{j_2}, z_{j_3}), \quad i = (i_1, i_2, i_3), j = (j_1, j_2, j_3),$$

Multi-index reordering:

$$a_{ij} = a_{(i_1, i_2, i_3)(j_1, j_2, j_3)} = a_{(i_1, j_1)(i_2, j_2)(i_3, j_3)} = a_{ijk}, \quad i = (i_1, j_1), j = (i_2, j_2), k = (i_3, j_3).$$

$\mathcal{A} = [a_{ijk}]$ is tensor now!

WHY TENSOR APPROXIMATION?

n	16	32	64	128	256
mem(A)	1.1Gb	72Gb	4.6Pb	288Pb	18Tb

WHY TENSOR APPROXIMATION?

n	16	32	64	128	256
mem(A)	1.1Gb	72Gb	4.6Pb	288Pb	18Tb

Can you buy a 18Tb hard disk?

WHY TENSOR APPROXIMATION?

n	16	32	64	128	256
mem(A)	1.1Gb	72Gb	4.6Pb	288Pb	18Tb

Tensor (approximate) decomposition for matrix:

$$A \approx \tilde{A}_r = \sum_{\alpha=1}^r \mathbf{U}_\alpha \times \mathbf{V}_\alpha \times \mathbf{W}_\alpha, \quad \mathbf{U}_\alpha = [u_{(i_1, j_1)\alpha}], \quad \mathbf{V}_\alpha = [v_{(i_2, j_2)\alpha}], \quad \mathbf{W}_\alpha = [w_{(j_3, j_3)\alpha}].$$

A has n^6 parameters, U, V, W have n^2r parameters

Superlinear compression? Fantastic!!

WHY TENSOR APPROXIMATION?

n	16	32	64	128	256
mem(A)	1.1Gb	72Gb	4.6Pb	288Pb	18Tb

Tensor (approximate) decomposition for matrix:

$$A \approx \tilde{A}_r = \sum_{\alpha=1}^r \mathbf{U}_\alpha \times \mathbf{V}_\alpha \times \mathbf{W}_\alpha, \quad \mathbf{U}_\alpha = [u_{(i_1, j_1)\alpha}], \quad \mathbf{V}_\alpha = [v_{(i_2, j_2)\alpha}], \quad \mathbf{W}_\alpha = [w_{(j_3, j_3)\alpha}].$$

A has n^6 parameters, U, V, W have n^2r parameters

Superlinear compression? Fantastic!!

Especially if $r \ll n$.

This is true for some classes of matrices, all integral equations belong to the class — you can try your data!

WHY TRILINEAR APPROXIMATION?

Tensor (approximate) decomposition for matrix:

$$A \approx \tilde{A}_r = \sum_{\alpha=1}^r \mathbf{U}_\alpha \times \mathbf{V}_\alpha \times \mathbf{W}_\alpha, \quad \mathbf{U}_\alpha = [\mathbf{u}_{(i_1, j_1)\alpha}], \mathbf{V}_\alpha = [\mathbf{v}_{(i_2, j_2)\alpha}], \mathbf{W}_\alpha = [\mathbf{w}_{(j_3, j_3)\alpha}].$$

Trilinear (approximate) decomposition for 3D-array (3D-tensor):

$$\mathcal{A} = [\mathbf{a}_{ijk}], \quad \mathbf{a}_{ijk} \approx \tilde{\mathbf{a}}_{ijk} = \sum_{\alpha=1}^r \mathbf{u}_{i\alpha} \mathbf{v}_{j\alpha} \mathbf{w}_{k\alpha}.$$

WHY TRILINEAR APPROXIMATION?

Tensor (approximate) decomposition for matrix:

$$A \approx \tilde{A}_r = \sum_{\alpha=1}^r \mathbf{u}_\alpha \times \mathbf{v}_\alpha \times \mathbf{w}_\alpha, \quad \mathbf{u}_\alpha = [\mathbf{u}_{(i_1, j_1)\alpha}], \quad \mathbf{v}_\alpha = [\mathbf{v}_{(i_2, j_2)\alpha}], \quad \mathbf{w}_\alpha = [\mathbf{w}_{(j_3, j_3)\alpha}].$$

Trilinear (approximate) decomposition for 3D-array (3D-tensor):

$$\mathcal{A} = [\mathbf{a}_{ijk}], \quad \mathbf{a}_{ijk} \approx \tilde{\mathbf{a}}_{ijk} = \sum_{\alpha=1}^r \mathbf{u}_{i\alpha} \mathbf{v}_{j\alpha} \mathbf{w}_{k\alpha}.$$

Tensor approximation \Leftrightarrow Trilinear approximation

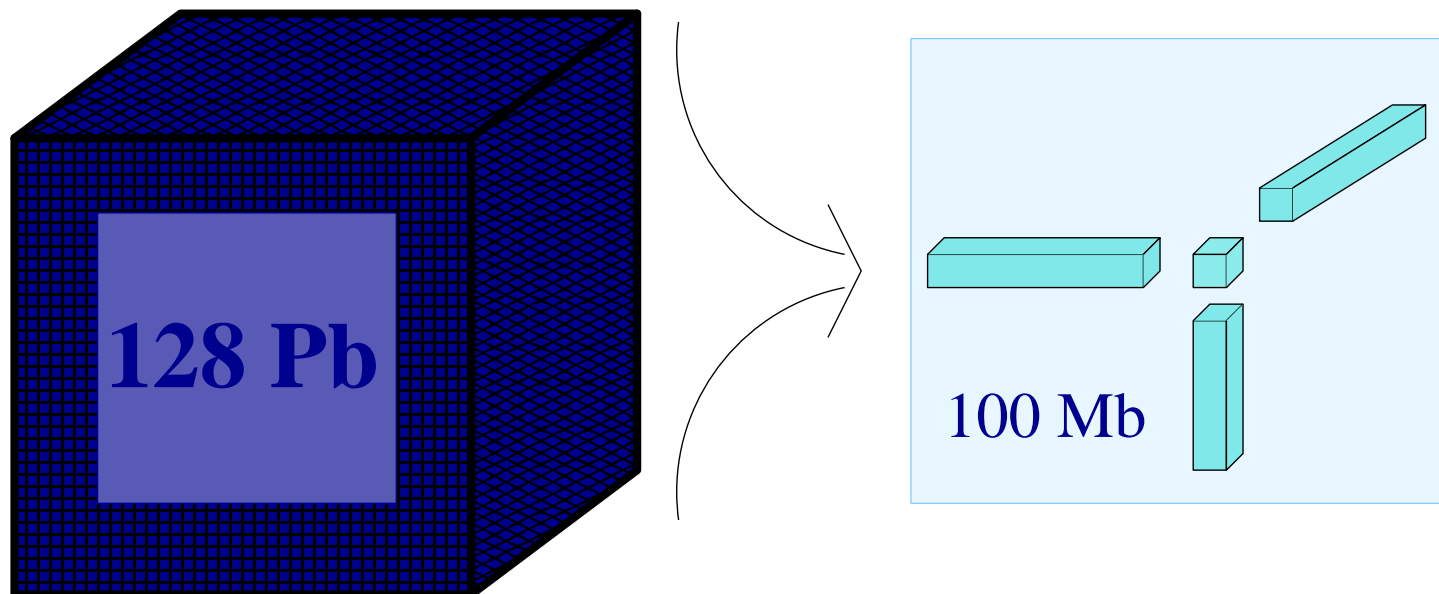
3L IS NOT ONLY FOR MATRICES

Multimedia applications:

- face recognition;
- speaker recognition;
- blind source separation;
- video/picture storage;
- image recognition.

Principal component analysis:

- statistics;
- economics;
- chemometrics;
- radiolocation;
- ...



TRILINEAR DECOMPOSITION

$$\mathbf{a}_{ijk} = \sum_{\alpha=1}^r \mathbf{u}_{i\alpha} \mathbf{v}_{j\alpha} \mathbf{w}_{k\alpha}.$$

MINIMISATION METHODS

$$\min_{\mathbf{u}, \mathbf{v}, \mathbf{w}} \sum_{i,j,k} \left(\sum_{\alpha=1}^r \mathbf{u}_{i\alpha} \mathbf{v}_{j\alpha} \mathbf{w}_{k\alpha} - \mathbf{a}_{ijk} \right)^2.$$

[+] standard minimisation algorithms: ALS, Gauss-Newton, Damped LM Newton.

[−] expensive iteration step, slow convergence.

[−] needs an beforehand knowledge of r .

Unapplicable if $n > 100$

TRILINEAR DECOMPOSITION

$$a_{ijk} = \sum_{\alpha=1}^r u_{i\alpha} v_{j\alpha} w_{k\alpha}.$$

MATRIX METHODS

- $r = n$ Simultaneous generalised Shur decomposition

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

[+] “Fast” method, that provide quite accurate first approximation

[−] Restricted to $r = n$.

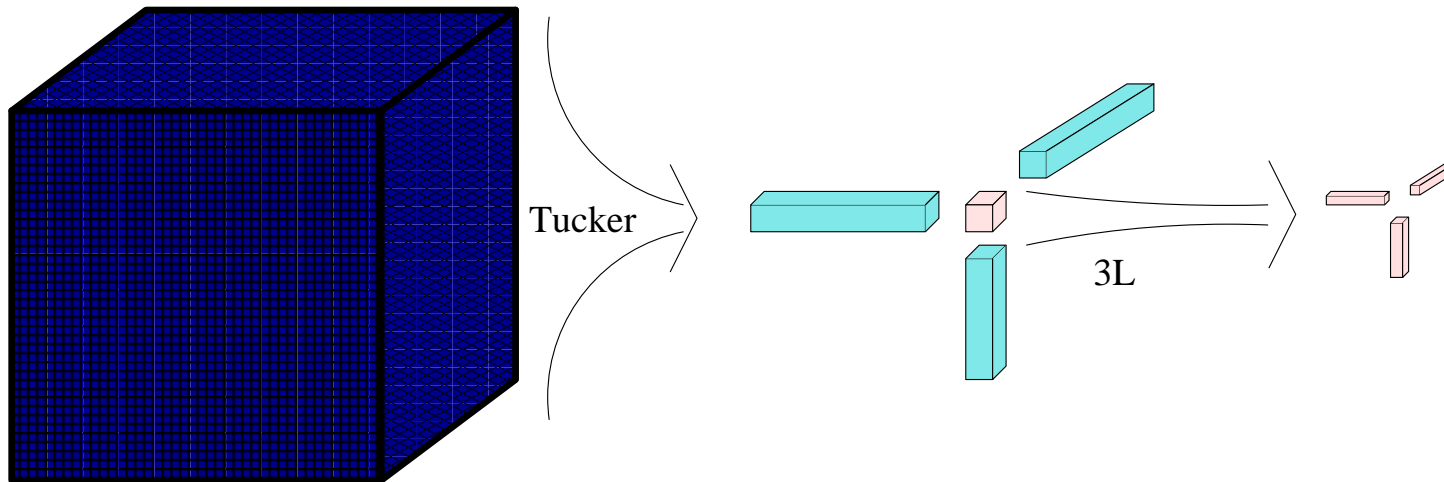
Unapplicable if $n > 100$

TUCKER DECOMPOSITION (performs dimensionality reduction)

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

factors U, V, W are orthogonal,
core tensor $\mathcal{G} = [g_{i'j'k'}]$ much smaller than \mathcal{A} .

TWO-STEP METHOD FOR LARGE n

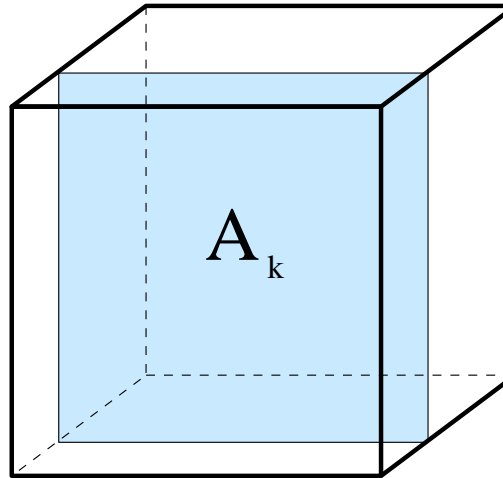


TUCKER DECOMPOSITION VIA 3 SVD

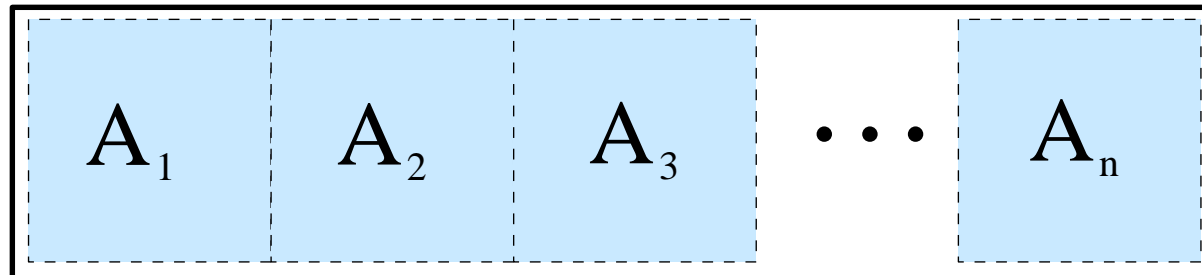
1. Unfold

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

Notation: slice



Notation: unfolding



TUCKER DECOMPOSITION VIA 3 SVD

1. Unfold

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

2. SVD

$$\mathbf{A}^{(1)} = \mathbf{U}\Sigma_1\Phi_1^\top, \quad \mathbf{A}^{(2)} = \mathbf{V}\Sigma_2\Phi_2^\top, \quad \mathbf{A}^{(3)} = \mathbf{W}\Sigma_3\Phi_3^\top,$$

TUCKER DECOMPOSITION VIA 3 SVD

1. Unfold

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

2. SVD

$$\mathbf{A}^{(1)} = \mathbf{U}\Sigma_1\Phi_1^\top, \quad \mathbf{A}^{(2)} = \mathbf{V}\Sigma_2\Phi_2^\top, \quad \mathbf{A}^{(3)} = \mathbf{W}\Sigma_3\Phi_3^\top,$$

3. Convolution

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

Cost $\mathcal{O}(n^4)$ flops plus n^3 evaluations of elements of \mathcal{A} .

TUCKER DECOMPOSITION VIA 3 SVD

1. Unfold

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

2. SVD

$$\mathbf{A}^{(1)} = \mathbf{U}\Sigma_1\Phi_1^\top, \quad \mathbf{A}^{(2)} = \mathbf{V}\Sigma_2\Phi_2^\top, \quad \mathbf{A}^{(3)} = \mathbf{W}\Sigma_3\Phi_3^\top,$$

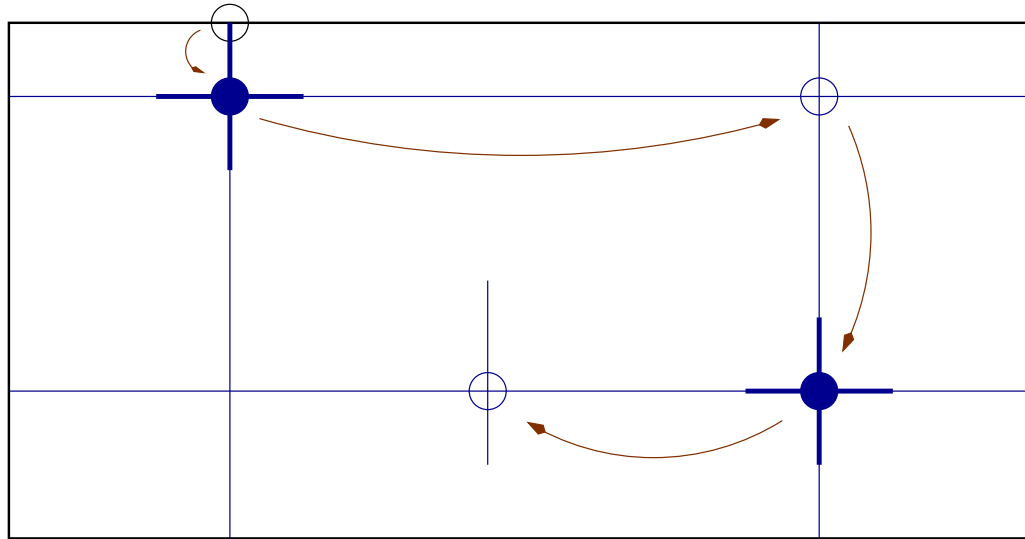
3. Convolution

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

Cost $\mathcal{O}(n^4)$ flops plus n^3 evaluations of elements of \mathcal{A} .

WE NEED FASTER METHOD (n^3 evaluations of elements are too expensive)

2D-CROSS METHOD (E. Tyrtyshnikov)



$$A \approx \tilde{A}_r = \sum_{q=1}^r u_q v_q^T.$$

0 $p = 1, j_1 = 1, \tilde{A}_0 = 0.$

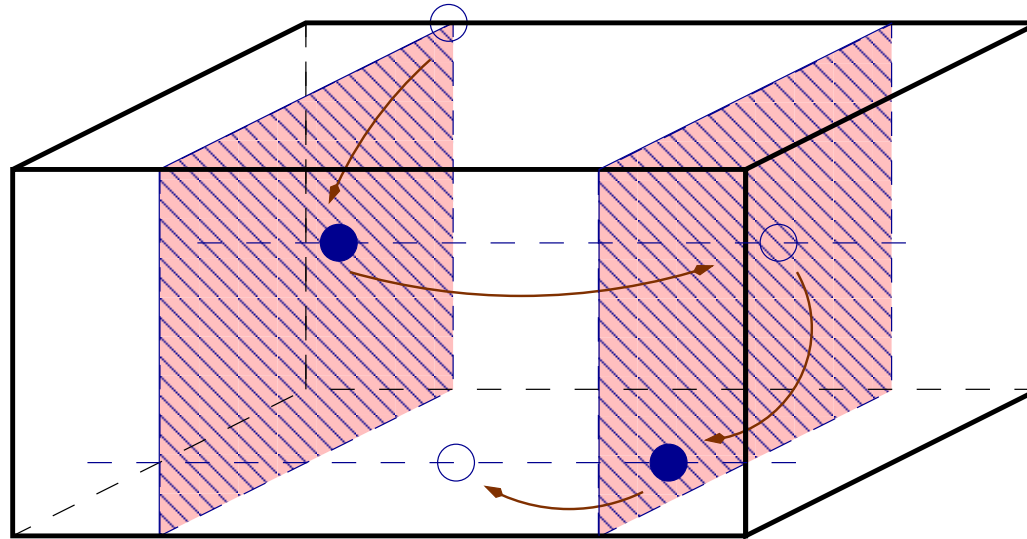
1 Compute column j_p of $A - \tilde{A}_p$. Find maximum element i_p .

2 Compute row i_p of $A - \tilde{A}_p$. Find maximum element $j_{p+1} \neq j_p$.

3 Calculate the new cross centered in (i_p, j_p) and set $\tilde{A}_p = \tilde{A}_{p-1} + u_p v_p^T$

4 If stopping criteria not satisfied, set $p := p + 1$ and goto 1.

TOWARDS 3D-CROSS METHOD ($\mathcal{O}(n^2)$)

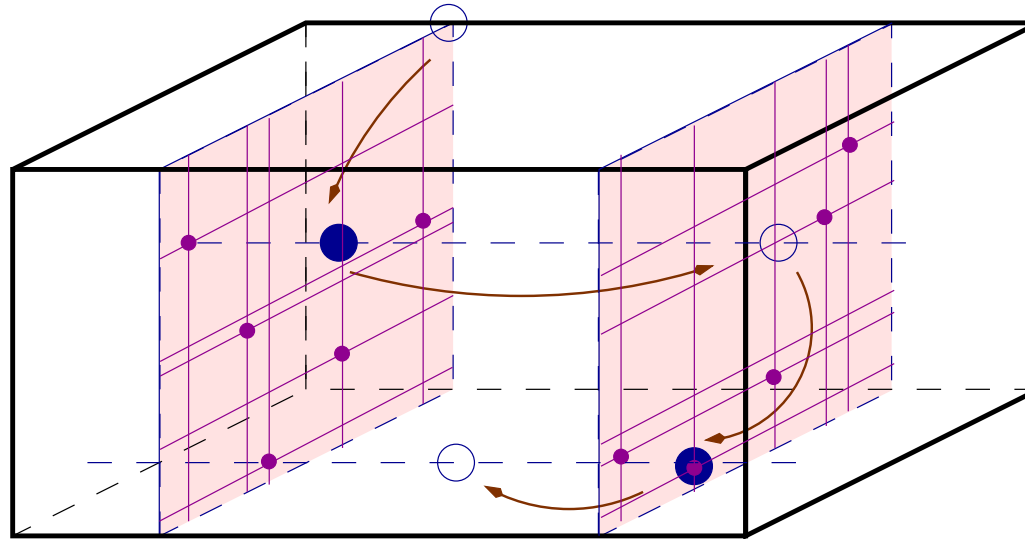


$$\mathcal{A} \approx \tilde{\mathcal{A}} = \sum_{q=1}^r \mathcal{A}_q \otimes \mathcal{W}_q.$$

Unfold \mathcal{A} and use 2D-Cross for $n \times n^2$ matrix.

(“long vector”) $v \rightarrow \mathcal{A}_{k_p}$ (slice)

IN LINEAR COMPLEXITY (nr^2 elements and $\mathcal{O}(nr^4)$ flops)



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

Unfold \mathcal{A} and use 2D-Cross for $n \times n^2$ matrix.

$$\text{("long vector")} \quad \mathbf{v} \quad \rightarrow \quad \mathbf{A}_{k,p} \quad (\text{slice})$$

Compute $\mathbf{A}_{k,p}$ with 2D-Cross method also.

SOME NUMERICAL EXAMPLES

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

RANKS AND ACCURACY

n	1.10 ⁻³		1.10 ⁻⁵		1.10 ⁻⁷		1.10 ⁻⁹	
	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

SOME NUMERICAL EXAMPLES

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

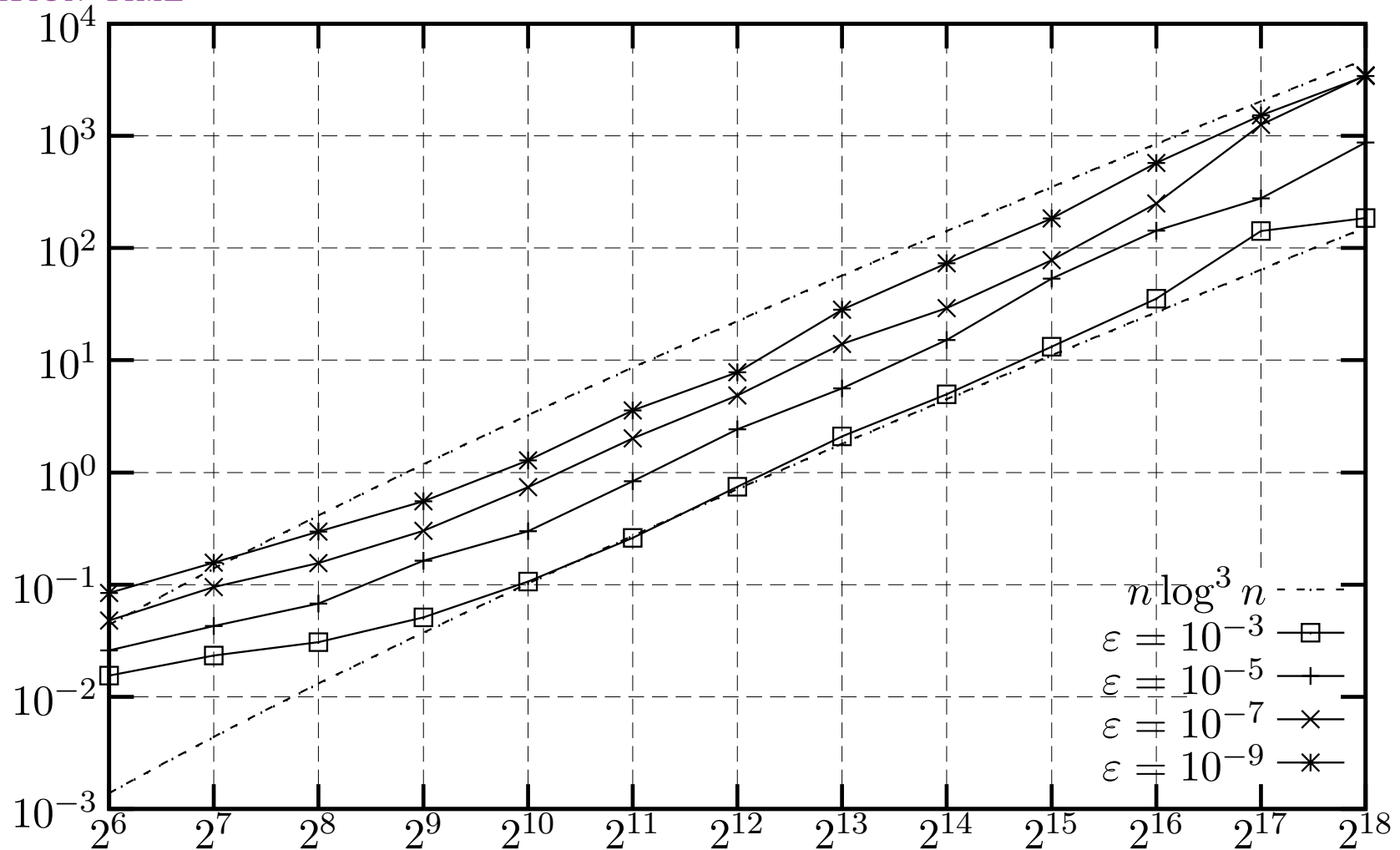
RANKS AND MEMORY

n	full	1.10 ⁻³		1.10 ⁻⁵		1.10 ⁻⁷		1.10 ⁻⁹	
		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

SOME NUMERICAL EXAMPLES

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

COMPUTATION TIME



VEM EQUATION EXAMPLE

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

$$Au = f, \quad u_{ijk}, f_{ijk} \text{ at nonuniform grid } n \times n \times n, \quad \varepsilon = 10^{-5}.$$

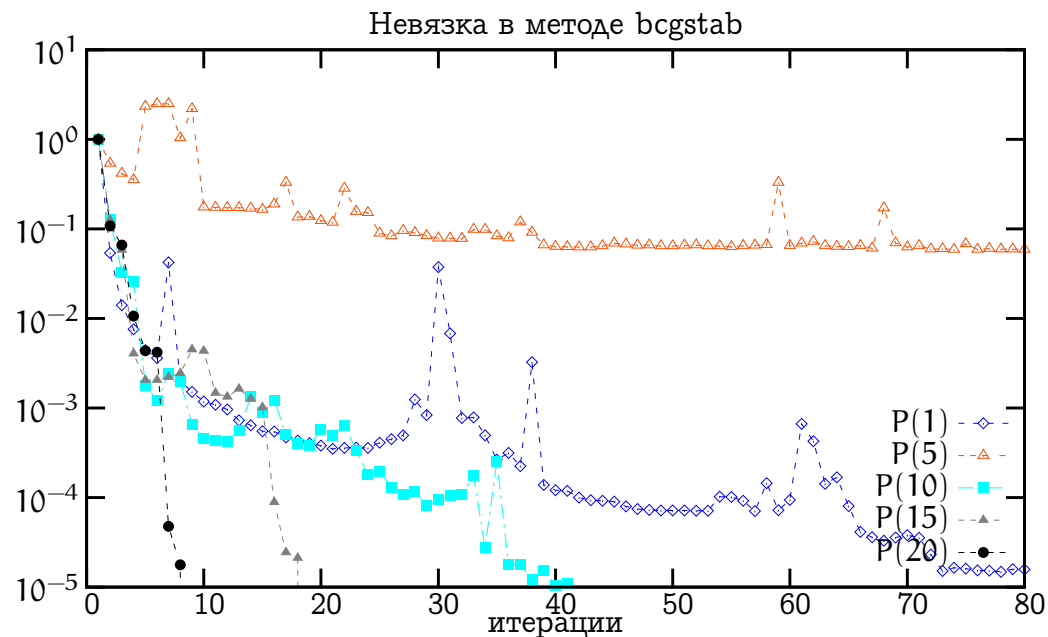
n	16	32	64	128	256	512
Full matrix	128MB	8GB	512GB	32TB	2PB	128PB
3L approx.	74KB	320KB	1.1MB	6MB	25MB	114MB
Rank	11	13	15	16	17	19
Time	0.6 sec.	1.5 sec.	8.4 sec.	54 sec.	5.5 min.	30 min.

VEM EQUATION EXAMPLE

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

$Au = f$, u_{ijk}, f_{ijk} at **nonuniform** grid $n \times n \times n$, $\varepsilon = 10^{-5}$.

n	16	32	64	128	256	512
Full matrix	128MB	8GB	512GB	32TB	2PB	128PB
3L approx.	74KB	320KB	1.1MB	6MB	25MB	114MB
Rank	11	13	15	16	17	19
Time	0.6 sec.	1.5 sec.	8.4 sec.	54 sec.	5.5 min.	30 min.



ELECTRON DENSITY APPROXIMATION

Knowing the “orbitals” for given molecule (CH_4 , C_2H_6 , $\text{C}_2\text{H}_5\text{OH}$), we compute electron density in n^3 points of the regular $n \times n \times n$ mesh with $n = 5121$. This is 1Tb of raw data.

The data are computed in 3L format with very large r :

$$\text{Rank}(\text{CH}_4) = 1334, \quad (160 \text{ Mb of data})$$

$$\text{Rank}(\text{C}_2\text{H}_6) = 3744, \quad (438 \text{ Mb of data})$$

$$\text{Rank}(\text{C}_2\text{H}_5\text{OH}) = 4212 \quad (500 \text{ Mb of data}).$$

Question: Can we approximate the data with better values of Rank and good accuracy?

ELECTRON DENSITY APPROXIMATION

Knowing the “orbitals” for given molecule (CH_4 , C_2H_6 , $\text{C}_2\text{H}_5\text{OH}$), we compute electron density in n^3 points of the regular $n \times n \times n$ mesh with $n = 5121$. This is 1Tb of raw data.

The data are computed in 3L format with very large r :

$$\text{Rank}(\text{CH}_4) = 1334, \quad (160 \text{ Mb of data})$$

$$\text{Rank}(\text{C}_2\text{H}_6) = 3744, \quad (438 \text{ Mb of data})$$

$$\text{Rank}(\text{C}_2\text{H}_5\text{OH}) = 4212 \quad (500 \text{ Mb of data}).$$

Question: Can we approximate the data with better values of Rank and good accuracy?

Answer: Yes. Approximation with $\varepsilon = 10^{-5}$:

$$\text{Rank}(\text{CH}_4) = 41 \quad (\text{Compression rate: 33, time: } < 30 \text{ minutes})$$

$$\text{Rank}(\text{C}_2\text{H}_6) = 62 \quad (\text{Compression rate: 60, time: about 1.5 hours}),$$

$$\text{Rank}(\text{C}_2\text{H}_5\text{OH}) = 68 \quad (\text{Compression rate: 62, time: about 2 hours}).$$

DETAILS

S.A. Goreinov, E. E. Tyrtyshnikov.

The maximum-volume concept in approximation by low-rank matrices //
Cont. Math., 2001.

I. V. Oseledets, D. V. Savostianov, E. E. Tyrtyshnikov.

Tucker dimensionality reduction of three-dimensional arrays in linear time //
SIMAX, 2008.

I. V. Oseledets, D. V. Savostianov.

Матричные методы и технологии решения больших задач
(yes, this preprint book is in Russian) //
INM RAS, 2005 — ask me, if you are interested.

Thank you for your attention!

Please ask questions.