TENSOR COMPUTATIONS: NEW FINDINGS AND PERSPECTIVES

E. E. Tyrtyshnikov

Institute of Numerical Mathematics of Russian Academy of Sciences

29 November 2010

WHAT ARE TENSORS?

Tensors = d-linear forms = d-dimensional arrays:

$$A = [a_{ij\ldots k}]$$

$$i \in I, \quad j \in J, \quad \ldots, \quad k \in K$$

Tensor A has:

- dimensionality (order) d = number of indices (modes, axes, directions, ways)
- size $n_1 \times \ldots \times n_d$ (number of nodes along each axis)

TENSORIZATION OF VECTORS AND MATRICES

One-dimensional arrays of size $N = n_1 \dots n_d$ can be represented as d-arrays (tensors):

$$x(i) = x(i_1 \dots i_d)$$

Matrices of size $N \times N$ naturally transform to $2d$-arrays:

$$a(i,j) = a(i_1 \dots i_d \,;\; j_1 \dots j_d)$$

$$i \leftrightarrow (i_1 \dots i_d)$$

$$j \leftrightarrow (j_1 \dots j_d)$$

Usually d can be chosen by many ways.

Why not to take d as large as possible?

Ultimate quantization means $n_1 = \dots = n_d = 2$:

$$d = \log_2 N$$

WHAT DOES HIGH-DIMENSIONAL MEAN?

NUMBER OF TENSOR ELEMENTS $= n^d$

GROWS EXPONENTIALLY IN d

HOW MANY AXES?

$2^{280} > 10^{83}$

(the number of atoms in the universe)

CURSE OF DIMENSIONALITY

HOW TO CONQUER THE CURSE OF DIMENSIONALITY?

- LOW-PARAMETRIC

  REPRESENTATIONS OF TENSORS

- METHODS OF COMPUTATIONS

  IN THESE REPRESENTATIONS

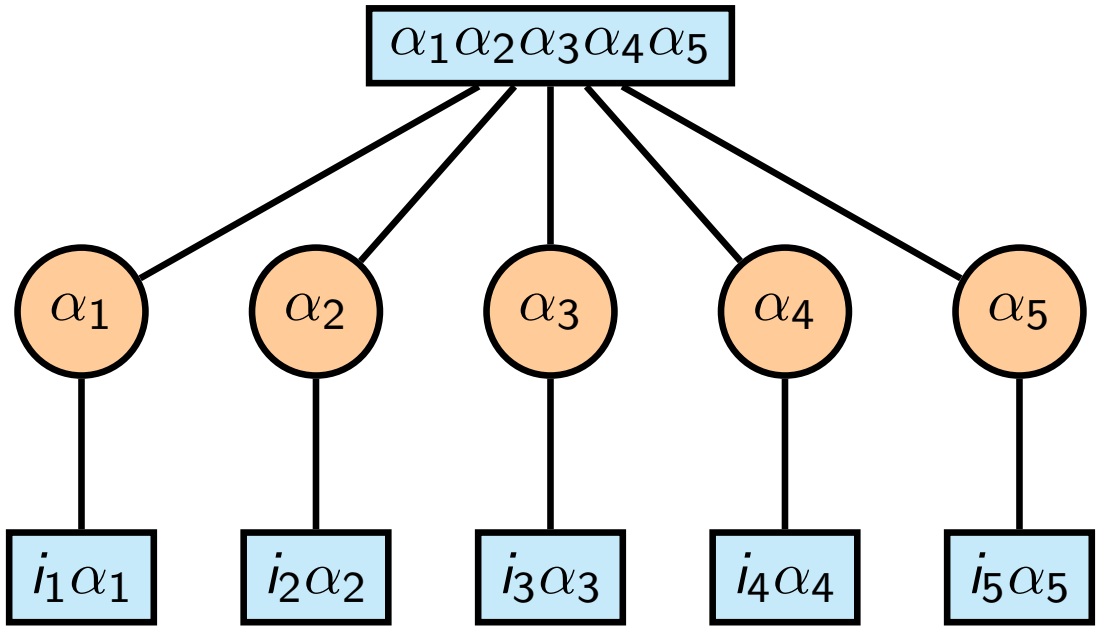
LINEAR DEPENDENCE ON THE NUMBER OF MODES d
WOULD CAUSE TO CONVERT VECTORS INTO d-TENSORS
WITH MAXIMAL POSSIBLE VALUE OF d

THEN, INSTEAD OF CURSE:

THE BLESSING OF DIMENSIONALITY!

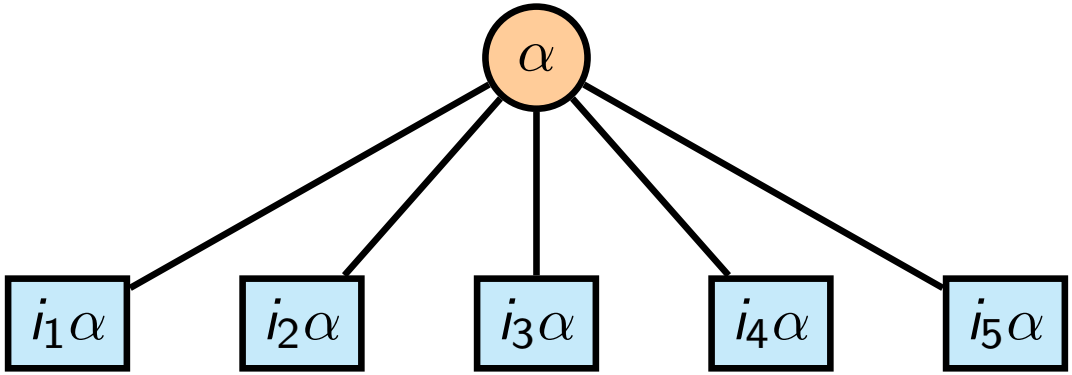
TUCKER DECOMPOSITION

$$a(i_1, ..., i_d) = \sum_{\alpha_1=1}^{r_1} ... \sum_{\alpha_d=1}^{r_d} g(\alpha_1, ..., \alpha_d)\ q_1(i_1, \alpha_1) ...\ q_d(i_d, \alpha_d)$$



CANONICAL POLYADIC DECOMPOSITION (CP, PARAFAC, CANDECOMP)

$$a(i_1, ..., i_d) = \sum_{\alpha=1}^{R} u_1(i_1, \alpha) ... \ u_d(i_d, \alpha)$$



TO BE CLEVER OR TO BE WISE?

CLEVER can find a way out of any difficulty,

WISE never comes to a difficulty.

RECURSIVE REDUCTION OF DIMENSIONALITY

Skeleton (dyadic) decomposition:

$$\begin{aligned} a(i_1, \ldots, i_d) &= a(i_1, \ldots, i_k;\ \ i_{k+1}, \ldots, i_d) \\ &= \sum_{\alpha=1}^{r} u(i_1, \ldots, i_k;\ \alpha)\ v(\alpha;\ i_{k+1}, \ldots, i_d) \end{aligned}$$

Dimensionality d reduces to dimensionalities $k+1$ and $d-k+1$.

Recursive (hierarchical) approach.

Central issue:

How do local approximation errors propagate?

RECURSIVE REDUCTION OF DIMENSIONALITY

FIRST PRESENTATIONS

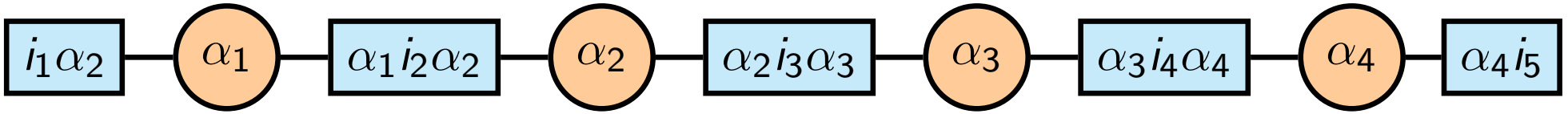
- W. Hackbusch, S. Kühn (MIS MPI)
  - L. Graesedyck (ENUMATH 2009, June 2009)
- I. Oseledets, E. Tyrtyshnikov (INM RAS)
  - GAMM Conference, Gdansk (February 2009)
  - Joint RFBR-DFG seminar, Berlin (May 2009)
  - ENUMATH 2009, Uppsala (June 2009)
  - Doklady RAS (2009), SIAM J. Sci. Comput. (2009)

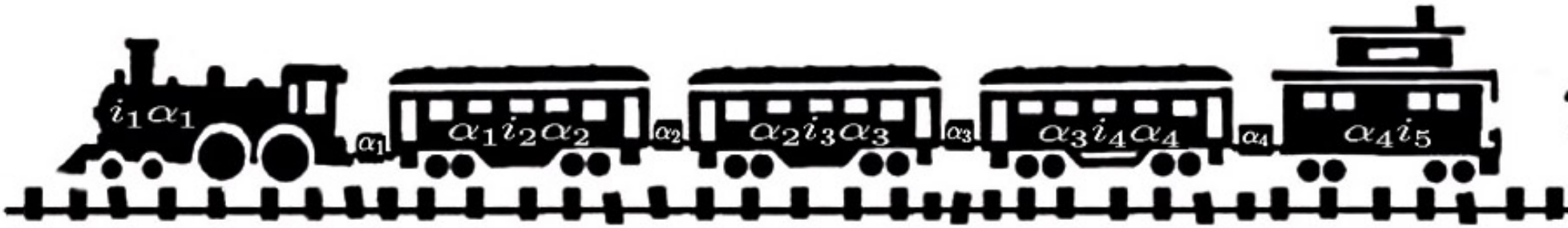
Any dimensionality reduction scheme reduces to *tensor trains*.

TENSOR-TRAIN DECOMPOSITION

$$a(i_1, ..., i_d) = \sum_{\alpha_1, ..., \alpha_{d-1}} g_1(i_1, \alpha_1)\, g_2(\alpha_1, i_2, \alpha_2) \ \dots$$

$$\dots \ g_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1})\, g_d(\alpha_{d-1}, i_d)$$





MATRIX PRODUCT STATE

$$a(i_1, \ldots, i_d) = G_1^{i_1} G_2^{i_2} \ldots G_{d-1}^{i_{d-1}} G_d^{i_d}$$

$$G_k^{i_k} \quad \text{is of size} \quad r_{k-1} \times r_k, \qquad r_0 = r_d = 1$$

$A(i_1, \ldots, i_d)$ is a number, $G_1^{i_1}$ is a row, $G_d^{i_d}$ is a column

TENSOR-TRAIN COMPONENTS

Tensor of size $n_1 \times \ldots \times n_d$

$$a(i_1, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \ldots \sum_{i_{d-1}=1}^{r_{d-1}}$$

$$g_1(i_1, \alpha_1) g_2(\alpha_1, i_2, \alpha_2) \ldots g_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) g_d(\alpha_{d-1}, i_d)$$

RANKS OF TENSOR CARRIAGES: $r_1, \ldots, r_{d-1}$

TT-RANK: $\sum_{i=1}^{d} n_i r_i^2 \; / \; \sum_{i=1}^{d} n_i$

$$\texttt{MEMORY} = (\texttt{SUM OF MODE SIZES}) \cdot (\texttt{TT-RANK})^2$$

THEOREM ON MINIMAL RANKS OF TT-DECOMPOSITION:

$$r_i \geqslant \operatorname{rank} A_i, \quad A_i = [a(i_1 \ldots i_k;\; i_{k+1} \ldots i_d)]$$

ADVANTAGES OF TENSOR TRAINS

INSTEAD OF

$$n^d$$

WE HAVE

$$dnr^2$$

In the case of ultimate quantization of vectors of size N:

$$n = 2, \quad d = \log_2 N$$

Since the complexity of tensor-train algorithms is linear in d, the complexity becomes logarithmic in N:

$$2r^2 \log_2 N$$

HOW TO PRODUCE A TENSOR TRAIN

Compute skeleton (dyadic) decompositions

$$a(i_1;\ i_2, i_3) = \sum_{\alpha_1} u_1(i_1, \alpha_1)\ v_1(\alpha_1; i_2, i_3)$$

$$v_1(\alpha_1, i_2;\ i_3) = \sum_{\alpha_2} u_2(\alpha_1, i_2;\ \alpha_2)\ v_2(\alpha_2, i_3)$$

It follows that

$$a(i_1, i_2, i_3) = \sum_{\alpha_1}\sum_{\alpha_2} u_1(i_1, \alpha_1)\ u_2(\alpha_1, i_2, \alpha_2)\ v_2(\alpha_2, i_3)$$

TT-SVD: $O(N \log_2 N)$ complexity with maximal quantization, and N elements are compressed to $2r^2 \log_2 N$ parameters.

EXAMPLES

$f(x)$ a function on $[0, 1]$

$$a(i_1, \ldots, i_d) = f(ih), \quad i = \frac{i_1}{2} + \frac{i_2}{2^2} + \cdots + \frac{i_d}{2^d}$$

Tensor of size $2 \times \cdots \times 2$.

EXAMPLE 1. $f(x) = e^x + e^{2x} + e^{3x}$
ttrank= 2.7 ERROR=1.5e-14

EXAMPLE 2. $f(x) = 1 + x + x^2 + x^3$
ttrank= 3.4 ERROR=2.4e-14

EXAMPLE 3. $f(x) = 1/(x - 0.1)$
ttrank= 10.1 ERROR=5.4e-14

THEOREM

If there exists an ε-approximation with separated variables

$$f(x+y) \approx \sum_{k=1}^{r} u_k(x) v_k(y), \quad r = r(\varepsilon),$$

then the vector of values of $f(x)$ admits a TT-approximation with error ε and ranks r.

If $f(x)$ is a sum of r exponents, then an exact TT-decomposition is with the carriages of rank r.

For a polynomial of degree m there is an exact TT-decomposition of rank $r = m + 1$.

For $f(x) = 1/(x - \delta)$ we have $r = \log \varepsilon^{-1} + \log \delta^{-1}$.

THREE MAIN ACHIEVEMENTS

- TENSOR-TRAIN ROUNDING
- TENSOR-TRAIN INTERPOLATION
- TENSOR-TRAIN WAVELET TRANSFORMS

TENSOR-TRAIN ARITHMETICS

Matrices and vectors are written as d-tensors in a tensor-train format (TT, QTT, WTT).

All results of operations (addition, matrix-by-matrix, matrix-by-vector) must be in the same format.

The key to computations is a **TENSOR-TRAIN ROUNDING**.

TENSOR-TRAIN ROUNDING

Given a tensor train with "large" carriages,

approximate it by another tensor train with "small" carriages.

Operation cost and memory must depend on d linearly!

- `DIRECT TT-ROUNDING`
  - TT-SVD ROUNDING (I. V. Oseledets)
    $O(dnr^3)$ `complexity!`
- `ITERATIVE TT-ROUNDING`
  - Krylov-Wedderburn method (S.Goreinov et al)
  - DMRG method (density matrix renormalization group).

Methods exploit *orthogonalization of tensor carriages*:

$$\sum_{\alpha_1, \ldots, \alpha_{d-1}} g_1(i_1;\, \alpha_1)\; g_2(\alpha_1, i_2;\, \alpha_2) \ldots\; g_k(\alpha_{k-1}, i_k; \alpha_k) \cdot$$

$$\cdot\; g_{k+1}(\alpha_k;\, i_{k+1}, \alpha_{k+1}) \ldots g_{d-1}(\alpha_{d-1};\, i_d)$$

NEW METHODS = OLD METHODS + TT-ROUNDING

EXAMPLE OF DISCRETE OPTIMIZATION:
EIGENVALUE PROBLEM FOR DIAGONAL MATRICES

Among a huge number of values of a function on a grid we need to find maximum or minimum. But we begin with a low-parametric representation of this array!

This is an eigenvalue problem:

$$Mx = \lambda x$$

M diagonal matrix of values of the function,
x eigenvector with 0 everywhere and 1 in exactly one place.

The difficulty is that the diagonal is never computed:
only its TT decomposition is given.

EXAMPLES FOR MAXIMUM

Block method of minimization of the Rayleigh quotient (A. V. Knyazev) with TT rounding in the block QTT format (O. S. Lebedeva).
Blocks of size 5, TT-ranks $\leqslant 5$.

Function	Domain	Size	Iter.	(Ax, x)	(Ae_i, e_i) $e_i \approx x$	Exact max
$\prod_{i=1}^{3}(1+0.1\,x_i+\sin x_i)$	$[1,50]^3$	2^{15}	30	428.2342	429.2342	429.2342
same	$[1,50]^3$	2^{30}	50	430.7838	430.7845	
$\prod_{i=1}^{3}(x_i+\sin x_i)$	$[1,20]^3$	2^{15}	30	8181.2	8181.2	8181.2
same	$[1,20]^3$	2^{30}	50	8181.2	8181.2	

EXAMPLES FOR MINIMUM

Function	Domain	Size	Iter.	(Ax, x)	(Ae_i, e_i) $e_i \approx x$	Exact min
$\prod_{i=1}^{3}(1+0.1\,x_i+\sin x_i)$	$[1,50]^3$	2^{15}	30	3.1389	3.1389	3.1389
same	$[1,50]^3$	2^{30}	50	2.5575	2.5544	
$\prod_{i=1}^{3}(x_i+\sin x_i)$	$[1,20]^3$	2^{15}	30	6.2445	6.2445	6.2445
same	$[1,20]^3$	2^{30}	50	6.2445	6.2445	
Rosenbrock: $(1-x)^2+100(y-x^2)^2$	$[-1,2]^2$	2^{20}	150	0.0252	0.0178	0.0064
same	$[-1,2]^2$	2^{30}	350	0.0134	0.0096	

Results of O. S. Lebedeva

TENSOR-TRAIN INTERPOLATION

Recover (approximate) a d-tensor

from a "small" part of its entries — of order of dnr^2,

where r is the maximal rank of tensor carriages.

GENERALIZATION OF SKELETON (DYADIC) DECOMPOSITION TO TENSORS (TT-cross)

I. Oseledets, E. Tyrtyshnikov: TT-cross approximation for multidimensional arrays, Linear Algebra Appl., 432 (2010), pp. 70-88.

HOW THIS PROBLEM IS SOLVED FOR MATRICES

Let A be close to a matrix of rank r:

$$\sigma_{r+1}(A) \leqslant \varepsilon$$

Then there exists a *cross* of r columns C and r rows R such that

$$|(A - CG^{-1}R)_{ij}| \leqslant (r+1)\varepsilon$$

G is an $r \times r$ matrix on the intersection of C and R
Take G of *maximal volume* among all $r \times r$ submatrices in A.

S.A.Goreinov, E.E.Tyrtyshnikov: The maximal-volume concept in approximation by low-rank matrices, *Contemporary Mathematics*, Vol. 208 (2001), 47-51.

S.A.Goreinov, E.E.Tyrtyshnikov, N.L.Zamarashkin: A theory of pseudo-skeleton approximations, *Linear Algebra Appl.* 261: 1-21 (1997). *Доклады РАН* (1995).

BEST IS AN ENEMY OF GOOD

Given A of size $n \times r$, find a row permutation to move a *good submatrix* to the upper $r \times r$ block. Since volume does not change by right-side multiplications, assume that

$$A = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ a_{r+1,1} & \dots & a_{r+1,r} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nr} \end{bmatrix}$$

NECESSARY: $|a_{ij}| \leqslant 1, \quad r+1 \leqslant i \leqslant n, \quad 1 \leqslant j \leqslant r$

Let this define a *good submatrix*. Then here is an ALGORITHM:

- If $|a_{ij}| \geqslant 1 + \delta$, then swap rows i and j.
- Make I in the first r rows by right-side multiplication.
- Quit if $|a_{ij}| \leqslant 1 + \delta$ for all i, j. Otherwise repeat.

MATRIX CROSS ALGORITHM

- Assume we are given some *initial* column indices $j_1, ..., j_r$.
- Find *maximal-volume* row indices $i_1, ..., i_r$ in these columns.
- Find *maximal-volume* column indices in the rows $i_1, ..., i_r$.
- Proceed choosing columns and rows until the skeleton cross approximations stabilize.

E.E.Tyrtyshnikov, Incomplete cross approximation in the mosaic-skeleton method, *Computing* 64, no. 4 (2000), 367–380.

CROSS TENSOR-TRAIN INTERPOLATION

Let $a_1 = a(i_1, i_2, i_3, i_4)$. Seek crosses in the unfolding matrices.
On input: r initial columns in each. Select *good* rows.

$$A_1 = [a(i_1\,;\ i_2, i_3, i_4)], \quad J_1 = \{i_2^{(\beta_1)} i_3^{(\beta_1)} i_4^{(\beta_1)}\}$$

$$A_2 = [a(i_1, i_2\,;\ i_3, i_4)], \quad J_2 = \{i_3^{(\beta_2)} i_4^{(\beta_2)}\}$$

$$A_3 = [a(i_1, i_2, i_3\,;\ i_4)], \quad J_3 = \{i_4^{(\beta_3)}\}$$

rows	matrix	skeleton decomposition
$I_1 = \{i_1^{(\alpha_1)}\}$	$a_1(i_1\,;\ i_2, i_3, i_4)$	$a_1 = \sum_{\alpha_1} g_1(i_1; \alpha_1)\ a_2(\alpha_1;\ i_2, i_3, i_4)$
$I_2 = \{i_1^{(\alpha_2)} i_2^{(\alpha_2)}\}$	$a_2(\alpha_1, i_2\,;\ i_3, i_4)$	$a_2 = \sum_{\alpha_2} g_2(\alpha_1, i_2;\ \alpha_2)\ a_3(\alpha_2, i_3;\ i_4)$
$I_3 = \{i_1^{(\alpha_3)} i_2^{(\alpha_3)} i_3^{(\alpha_3)}\}$	$a_3(\alpha_2, i_3\,;\ i_4)$	$a_3 = \sum_{\alpha_3} g_3(\alpha_2, i_3;\ \alpha_3)\ g_4(\alpha_3;\ i_4)$

Finally

$$a = \sum_{\alpha_1,\alpha_2,\alpha_3,\alpha_4} g_1(i_1, \alpha_1)\ g_2(\alpha_1, i_2, \alpha_2)\ g_3(\alpha_2, i_3, \alpha_3)\ g_4(\alpha_3, i_4)$$

TT-CROSS IMPLEMENTATIONS

- I. V. Oseledets (April 2009)
- G. Bonik, V. Rumyantsev
  (diploma work, 2009)
- D. V. Savostyanov (October 2009)
  DMRG-based, coined during a train journey in Italy

TENSOR-TRAIN QUADRATURES

Example (G. Bonik)

$\int_0^\infty \frac{\sin x}{x} dx = \frac{\pi}{2}$.

Strange idea: exploit the rule of rectangles.

Required number of nodes: 2^{77}.

QTT-ranks: 12, time: 1 sec.

QTT is a convenient parametrization for functions!

COMPUTE d-DIMENSIONAL INTEGRALS

$$I(d) = \int \sin(x_1 + x_2 + \ldots + x_d)\, dx_1 dx_2 \ldots dx_d =$$

$$\mathrm{Im} \int_{[0,1]^d} e^{i(x_1+x_2+\ldots+x_d)}\, dx_1 dx_2 \ldots dx_d = \mathrm{Im}((\frac{e^i - 1}{i})^d)$$

Use the Chebyshev (Clenshaw-Curtis) quadrature with $n = 11$ nodes. All n^d values are NEVER COMPUTED! Instead, we find a TT cross and construct a TT approximation for this tensor.

d	I	Relative accuracy	Time
10	-6.299353e-01	1.409952e-15	0.14
100	-3.926795e-03	2.915654e-13	0.77
500	-7.287664e-10	2.370536e-12	4.64
1000	-2.637513e-19	3.482065e-11	11.60
2000	2.628834e-37	8.905594e-12	33.05
4000	9.400335e-74	2.284085e-10	105.49

TENSOR TRAINS IN QUANTUM MOLECULAR DYNAMICS

Joint German-Russian Laboratory GERRUS-LAB (Moscow-Leipzig)
www.inm.ras.ru/gerrus

Molcecular Shrōdinger equation

$$H\Psi = (-\frac{1}{2}\Delta + V(R_1, \ldots, R_f))\Psi = E\Psi$$

V given potential

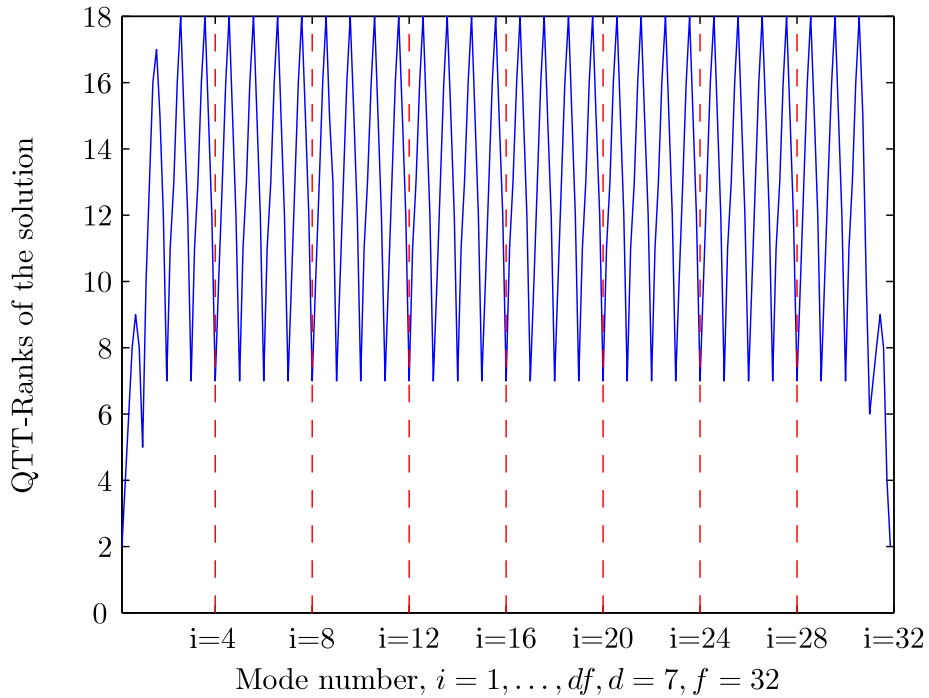
EXAMPLE:

$V(q_1, \ldots, q_f) = \frac{1}{2}\sum_{k=1}^{f} q_k^2 + \lambda \sum_{k=1}^{f-1} \left(q_k^2 q_{k+1} - \frac{1}{3} q_k^3\right).$

Henon-Heiles potential.

TENSOR TRAINS IN QUANTUM MOLECULAR DYNAMICS

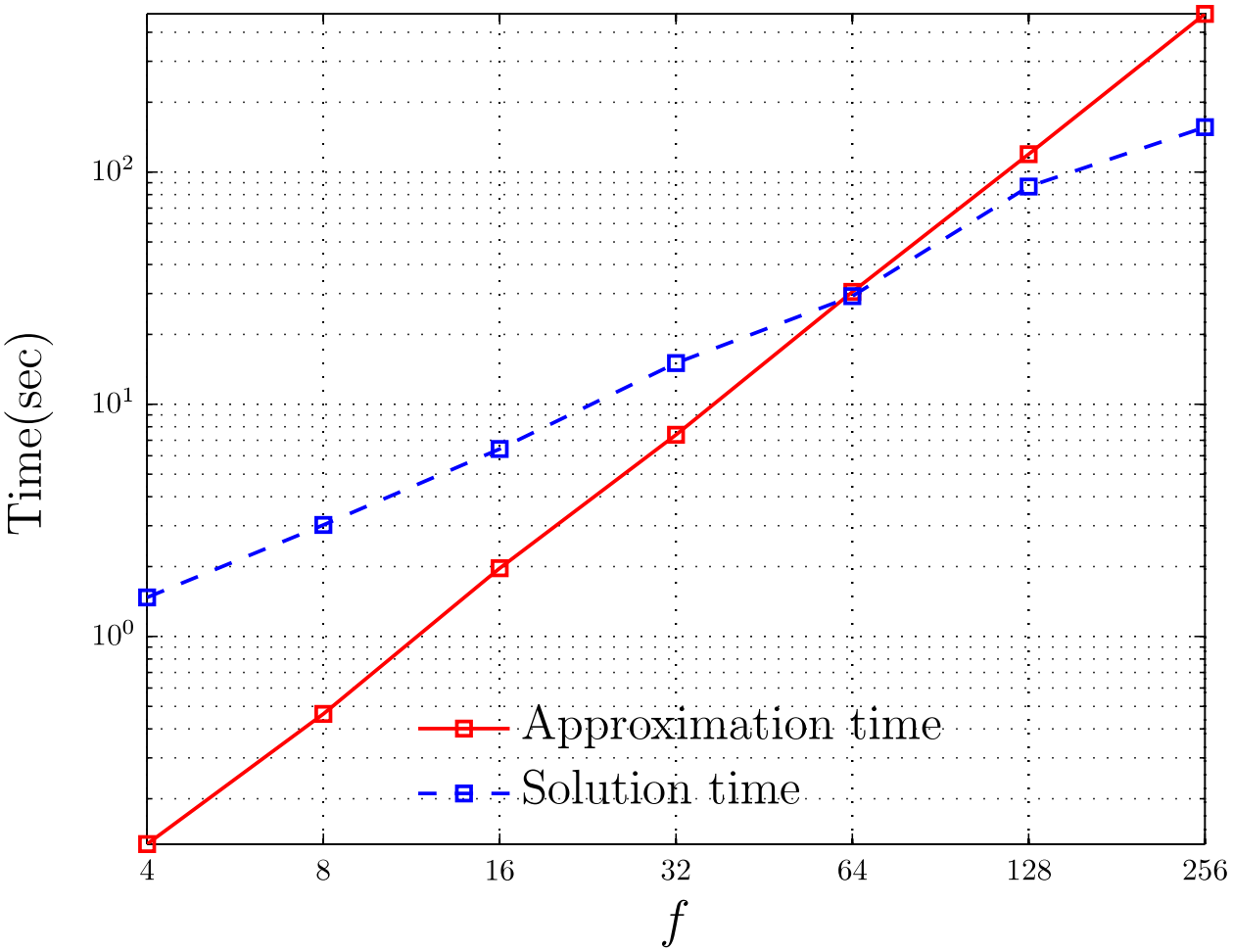
$$V(q_1, \ldots, q_f) = \frac{1}{2}\sum_{k=1}^{f} q_k^2 + \lambda \sum_{k=1}^{f-1} \left(q_k^2 q_{k+1} - \frac{1}{3} q_k^3\right).$$



QTT-ranks of the Henon-Heiles eigenfunction‘
(I. V. Oseledets, B. N. Khoromskij)

TENSOR TRAINS IN QUANTUM MOLECULAR DYNAMICS

Timings for the Henon-Heiles problem (I. V. Oseledets)



FROM TENSOR TRAINS TO ALGEBRAIC WAVELETS

$$a(i_1;\, i_2, i_3) = \sum_{\alpha_1=1}^{r} u_1(i_1, \alpha_1)\; v_1(\alpha_1; i_2, i_3) + \texttt{ERROR}$$

In TT we choose r so that ERROR can be *neglected*.

What if we just *fix* r?

To keep the accuracy we may need to add some largest entries from ERROR.

Hopefully it is a small enough set, which means *pseudo-sparsity* of ERROR.

In this way **TT** becomes **WTT**.

FROM TENSOR TRAINS TO ALGEBRAIC WAVELETS

$$a(i_1;\, i_2, i_3, \ldots) = \sum_{\alpha_1=1}^{r} u_1(i_1, \alpha_1)\; v_1(\alpha_1; i_2, i_3, \ldots) + \texttt{ERROR}$$

Using SVD we construct the best rank-r approximation.

FIRST STEP: Let u_1 have orthonormal columns.
Then v_1 is a unitary transformation (*filtering*) of a.

SECOND STEP: Do the same with a smaller array
$v_1(\alpha_1 i_2;\, i_3, \ldots)$.

REPEAT RECURSIVELY.

FROM TENSOR TRAINS TO ALGEBRAIC WAVELETS

FILTER BANKS $u_1, u_2, \ldots$ ARE ADAPTED TO A SIGNAL

AND CONSTRUCTED BY PURELY ALGEBRAIC TOOLS

FROM TENSOR TRAINS TO ALGEBRAIC WAVELETS

FIRST STAGE:

Computation of WTT-filters.
It may be done for one representative signal from some family.
E.g. for $f(x) = x^k$. Then only $k + 1$ nonzeros in the image.

SECOND STAGE:

Computation of the WTT-image
(linear transformation with WTT-filters).

THIRD STAGE:

Sparsification of the WTT-image.

FROM TENSOR TRAINS TO ALGEBRAIC WAVELETS

Consider a signal on a uniform grid

$$f(x) = \sin(100x),\ x \in [0, 1].$$

All TT-ranks are exactly equal to 2, whereas Daubechies transforms leave large amount of nonzero elements.

ε	nnz(WTT)	Memory(U_k)	nnz(D4)	nnz(D8)
10^{-4}	2	152	3338	880
10^{-6}	2	152	19696	2010
10^{-8}	2	152	117575	6570
10^{-10}	2	152	845869	15703
10^{-12}	2	152	1046647	49761

Sine function, $n = 2^d$, $d = 20$

WTT FOR MATRICES

Compute $\widehat{A} = WAW^\top$, where W is a suitable wavelet transform. Take

$$A_{ij} = \begin{cases} \frac{1}{i-j}, & i \neq j \\ 0, & i = j \end{cases}, \quad 1 \leq i, j \leq 2^d.$$

ε	nnz(WTT(x))	nnz(WTT(x^3))	nnz(D4)	nnz(D8)
10^{-4}	43732	46546	54002	35456
10^{-6}	95898	71356	194188	66740
10^{-8}	196518	116238	588914	141794
10^{-10}	376378	182716	814022	291732
10^{-12}	623212	283972	926328	549962

Comparison of WTT and Daubechies transforms

$$d = 10$$

TENSOR TRAINS FOR MATRICES

If an $N \times N$-matrix with elements $a(i,j)$ is nonsingular, then a TT-decomposition for the tensor $a(i_1 \ldots i_d; j_1 \ldots j_d)$ has a carriage of rank N. Very bad!

This is cured if we get to a *transposed* tensor

$$b(i_1 j_1,\ i_2 j_2,\ \ldots\ , i_d j_d) = a(i_1 \ldots i_d; j_1 \ldots j_d)$$

EXAMPLE. $a(i,j) = 1/(i - j + 0.5)$

(Hilbert, Cauchy, C. Moler, I. K. Lifanov)

```
N=1024    ttrank= 11.4    ERROR=1.7e-13
N=2048    ttrank= 12.0    ERROR=1.0e-12
N=4096    ttrank= 12.4    ERROR=4.7e-12
```

WWT FOR RESHAPED VECTORIZED MATRICES

$$A_{ij} = \begin{cases} \frac{1}{i-j}, & i \neq j \\ 0, & i = j \end{cases}, \quad 1 \leq i, j \leq 2^d.$$

$n = 2^d$	Memory(WTT)	nnz(D4)	nnz(D8)	nnz(D20)
2^5	388	992	992	992
2^6	752	4032	3792	3348
2^7	1220	15750	13246	8662
2^8	1776	59470	41508	20970
2^9	2260	213392	102078	45638
2^{10}	2744	780590	215738	95754
2^{11}	3156	1538944	306880	176130

Comparison of 2D WTT and Daubechies transforms

$$\varepsilon = 10^{-8}$$

KAZEEV PRODUCT

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \bowtie \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} =$$

$$\begin{bmatrix} A_{11} \otimes B_{11} + A_{12} \otimes B_{21} & A_{11} \otimes B_{12} + A_{12} \otimes B_{22} \\ A_{21} \otimes B_{11} + A_{22} \otimes B_{21} & A_{21} \otimes B_{12} + A_{22} \otimes B_{22} \end{bmatrix},$$

$$\boldsymbol{A = G_1 \bowtie G_2 \bowtie \ldots \bowtie G_d}$$

$\boldsymbol{G_k}$ is viewed as a block $\boldsymbol{r_{k-1} \times r_k}$ matrix with $\mathbf{2 \times 2}$ blocks

Convenient to derive analytical TT-decompositions!

KAZEEV PRODUCT

One-dimensional Laplace matrix of order $n = 2^d$
Dirichlet boundary conditions

$$\Delta = \begin{bmatrix} I & J^\top & J \end{bmatrix} \bowtie \begin{bmatrix} I & J^\top & J \\ & J & \\ & & J^\top \end{bmatrix}^{\bowtie(d-2)} \bowtie \begin{bmatrix} 2I - J - J^\top \\ -J \\ -J^\top \end{bmatrix}$$

B. N. Khoromskij, V. A. Kazeev (GERRUS-LAB)

CONCLUSIONS AND PERSPECTIVES

- TT-decompositions and ôô-algorithms (see `http://pub.inm.ras.ru`) are effecient instruments for comspresison of vectors and matrices. TT-toolbox Matlab: `http://spring.inm.ras.ru/osel`.
- Memory and operation cost depend of matrix size logarithmically.
- New methods of tabulation of functions.
- Possibilities of accurate and fast computation of integrals based on TT-interpolation.
- Applications to image and signal processing.
- Applications to spectral problems of quantum chemistry, quantum molecular dynamics, optimization by parameters (model reduction), stochastic differential equations.