# Numerical tensor methods and their applications

I.V. Oseledets

8 May 2013

# All lectures

4 lectures,

- 2 May, 08:00 - 10:00: Introduction: ideas, matrix results, history.
- 7 May, 08:00 - 10:00: Novel tensor formats (TT, HT, QTT).
- 8 May, 08:00 - 10:00: Advanced tensor methods (eigenproblems, linear systems).
- 14 May, 08:00 - 10:00: Advanced topics, recent results and open problems.

# Brief recap of Lecture 2

- Tensor Train format
- Arithmetics
- Rounding
- QTT-format (idea of tensorization)
- Cross approximation

# Plan of lecture 2

- QTT-format: explicit representations of functions
- QTT-format: explicit representation of operators
- Classification theory
- QTT-Fourier transform
- QTT-convolution
- Linear systems
- Eigenvalue problems

# QTT-format (publications-first)


[1] S. V. Dolgov, B. N. Khoromskij, and D. V. Savostyanov. Superfast Fourier transform using QTT approximation. *J. Fourier Anal. Appl.*, 18(5):915–953, 2012.

[2] V. Kazeev, B. N. Khoromskij, and E. E. Tyrtyshnikov. Multilevel Toeplitz matrices generated by tensor-structured vectors and convolution with logarithmic complexity. Technical Report 36, MPI MIS, Leipzig, 2011.

[3] V. A. Kazeev and B. N. Khoromskij. Low-rank explicit QTT representation of the Laplace operator and its inverse. *SIAM J. Matrix Anal. Appl.*, 33(3):742–758, 2012.

[4] B. N. Khoromskij. $\mathcal{O}(d \log n)$–Quantics approximation of $N$–$d$ tensors in high-dimensional numerical modeling. *Constr. Appr.*, 34(2):257–280, 2011.

[5] I. V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM J. Matrix Anal. Appl.*, 31(4):2130–2145, 2010.

# QTT-format

We have a vector $v$ of values of a function $f$ on a uniform grid with $2^d$ points:

$$v(i) = f(x_i), \quad x_i = a + ih, \quad h = (b-a)/(n-1).$$

# QTT-format

We have a vector $v$ of values of a function $f$ on a uniform grid with $2^d$ points:

$$v(i) = f(x_i), \quad x_i = a + ih, \quad h = (b-a)/(n-1).$$

Reshaping into a tensor:

$$i \longrightarrow (i_0, i_1, \ldots, i_{d-1}).$$

$$i = i_1 + 2i_2 + 4i_3 + \ldots + 2^{d-1} i_d$$

$$V(i_1, \ldots, i_d) = v(i).$$

# QTT-format

Finally, we have:

$$V(i_1, \ldots, i_d) = f(t_1 + \ldots + t_d), \quad t_k = \frac{a}{d} + 2^k i_k h$$

# Exponential function

$$f(x) = \exp \lambda x,$$

Then $f(t_1 + \ldots + t_d) = \exp(\lambda t_1) \ldots \exp(\lambda t_d)$, it has rank 1!

# Linear function

$$f(x) = x$$

$$f(t_1 + \ldots + t_d) = t_1 + \ldots + t_d$$

# Linear function(full steps)

$$t_1 + t_2 + t_3 + t_4 =$$

# Linear function(full steps)

$$t_1 + t_2 + t_3 + t_4 = \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_2 + t_3 + t_4 \end{pmatrix}$$

# Linear function(full steps)

$$t_1 + t_2 + t_3 + t_4 = \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_2 + t_3 + t_4 \end{pmatrix} =$$

$$= \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_3 + t_4 \end{pmatrix}$$

# Linear function(full steps)

$$t_1 + t_2 + t_3 + t_4 = \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_2 + t_3 + t_4 \end{pmatrix} =$$

$$= \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_2 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_3 + t_4 \end{pmatrix} =$$

$$= \begin{pmatrix} t_1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_3 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t_4 \end{pmatrix}$$

# Sine function

Similar representation can be obtained for a sine function:

$$f(x) = \sin \lambda x$$

$$f(t_1 + \ldots + t_d) = \sin(t_1 + \ldots + t_d) =$$

$$= \begin{pmatrix} \sin t_1 & \cos t_1 \end{pmatrix} \begin{pmatrix} \sin t_2 & -\cos t_2 \\ \cos t_2 & \sin t_2 \end{pmatrix} \ldots \begin{pmatrix} \sin t_{d-1} & -\cos t_{d-1} \\ \cos t_{d-1} & \sin t_{d-1} \end{pmatrix} \begin{pmatrix} \cos x_d \\ \sin x_d \end{pmatrix}$$

The rank is still 2!

# General result

**Theorem**

Let $f$ be such that (O. Const. Approx., 2013)

$$f(x+y) = \sum_{\alpha=1}^{r} u_\alpha(x) v_\alpha(y)$$

then the QTT-ranks are bounded by $r$

Interesting example: rational functions

# TT-format for matrices

What about matrices?

# TT-format for matrices

What about matrices?

Solution - a vector $x$ associated with a $d$-tensor $X(i_1, \ldots, i_d)$

Linear operators, acting on such tensors, can be indexed as

$$A(i_1, \ldots, i_d; j_1, \ldots, j_d).$$

Terminology: $d$-level matrix

# Two-level matrix: illustration

Even in 2d it is interesting: $A, B$ are $n \times n$,
Kronecker product

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & a_{13}B & a_{14}B \\ a_{21}B & a_{22}B & a_{23}B & a_{24}B \\ a_{31}B & a_{32}B & a_{33}B & a_{34}B \\ a_{41}B & a_{42}B & a_{43}B & a_{44}B \end{pmatrix}$$

# Two-level matrix: illustration

In the index form:

$$C = A \otimes B$$

$$C(i_1, i_2; j_1 j_2) = A(i_1, j_1) B(i_2, j_2)$$

# Two-level matrix: illustration

In the index form:

$$C = A \otimes B$$

$$C(i_1, i_2; j_1 j_2) = A(i_1, j_1) B(i_2, j_2)$$

Exactly rank-1 decomposition under permutation

# Back to $d$-dimensions

Let $A$ be a d-level matrix: $A(i_1, \ldots, i_d; j_1, \ldots, j_d)$
(say, $d$-dimensional Laplace operator)

# Back to $d$-dimensions

Let $A$ be a d-level matrix: $A(i_1, \dots, i_d; j_1, \dots, j_d)$

(say, $d$-dimensional Laplace operator)

No low TT-ranks if considered as a $2d$-array!

# Back to $d$-dimensions

Right way: permute indices

$$B(i_1 j_1; i_2 j_2; \ldots i_d j_d) = A(i_1, \ldots, i_d; j_1, \ldots, j_d)$$

$$A(i_1, \ldots, i_d; j_1, \ldots, j_d) = A_1(i_1, j_1) A_2(i_2, j_2) \ldots A_d(i_d, j_d)$$

# Matrix-by-vector product

$$A(i_1, \ldots, i_d; j_1, \ldots j_d) = A_1(i_1, j_1) \ldots A_d(i_d, j_d)$$

$$X(j_1, \ldots, j_d) = X_1(j_1) \ldots X_d(j_d)$$

$$Y(I) = \sum_J A(I, J) X(J)$$

Exercise: Find a formula

# QTT-matrix

Matrices in the QTT-format:

$$a_{ij} = \frac{1}{i - j + 0.5}, \quad i, j = 1, \dots, 2^d.$$

Let us see what are the ranks. (Demo)

# Laplace operator

Consider an operator $-\frac{d^2}{dx^2}$ with Dirichlet boundary conditions, discretized using the simplest finite-difference scheme:

(Illustration for $n = 4$)

$$\Delta = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

(Demo)

# Laplace operator

V. Kazeev: QTT-ranks of the Laplace operator are bounded by 3

# Laplace operator: general results

We will need a special matrix-by-matrix product:

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

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

Doing Kronecker product for the blocks!

# Basic QTT-blocks

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$I_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad I_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

$$E = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad F = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix},$$

$$K = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad L = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

# QTT representation of 1D-Laplace

$$\boldsymbol{\Delta}_{DD}^{(d)} = \begin{bmatrix} I & J' & J \end{bmatrix} \bowtie \begin{bmatrix} I & J' & J \\ & J & \\ & & J' \end{bmatrix}^{\bowtie (d-2)} \bowtie \begin{bmatrix} 2I - J - J' \\ -J \\ -J' \end{bmatrix}$$

# QTT representation in the Toolbox

In the TT-Toolbox, it is defined via the function `tt_qlaplace_dd`

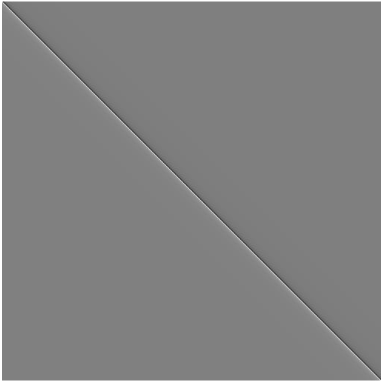
# Wavelets and tensor trains

QTT can be applied to matrices:

$$A(i, j) \longrightarrow A(i_1, \ldots, i_d, j_1, \ldots, j_d) \longrightarrow A(i_1, j_1, i_2, j_2, \ldots, i_d, j_d)$$

# Wavelets and tensor trains

Smooth function and/or special function:
$512 \times 512$ "Hilbert image": $a_{ij} = 1.0/(i - j + 0.5)$

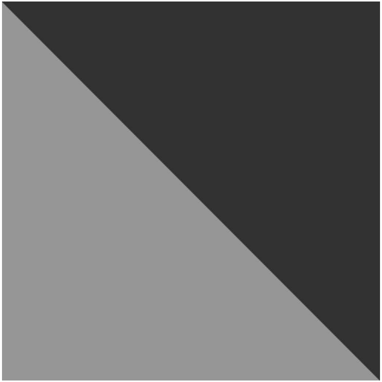


540 bytes (QTT) vs 8 KB (JPEG)

# Wavelets and tensor trains

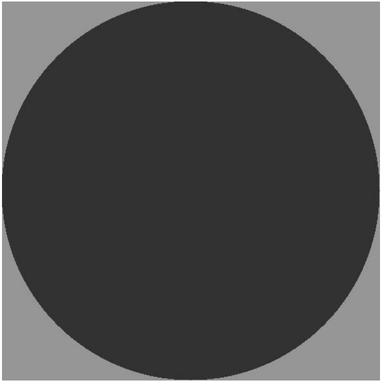
QTT compression of simple images

Good: Triangle



50 bytes (QTT) vs 8 KB (JPEG)

BAD: Circle



70 KB (QTT) vs 8 KB (JPEG)

# Wavelets and tensor trains

Wavelet tensor train:
One step of TT-SVD is equivalent to:

$$U^\top A = \begin{pmatrix} v_{11} & v_{12} & \dots \\ v_{21} & v_{22} & \dots \\ v_{31} & v_{32} & \dots \\ v_{41} & v_{42} & \dots \end{pmatrix}$$

# Problems with a circle

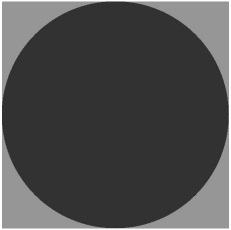
Wavelet: First (dominant) rows compress further, others are sparse

QTT: Leave only rows of large norms (large singular values)

# Problems with a circle

That is why it is bad:

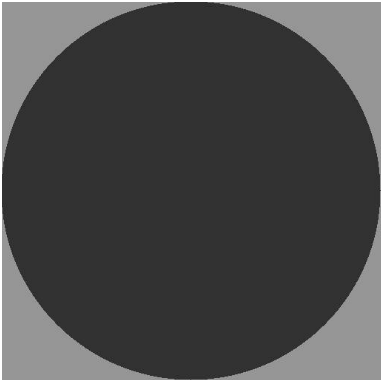
$$r \sim n, \longrightarrow mem = \mathcal{O}(n^2)$$



# Idea with the circle

Leaving sparse singular vectors — a novel digital data compression technique (to be combined with others)! Wavelet decomposition with adaptive number of moments!

# Image compression



New: 50 bytes (WTT) vs 8 KB (JPEG)

# Image compression

latex

Lena

7 KB(WTT), PSNR 32.45

12 KB (WTT), PSNR 35.62

19 KB (WTT), PSNR 38.79

# Fourier transform

Consider now an important operation: the Fourier transform (FFT)

Example of a matrix with large QTT-ranks! (We can test)

$$y = Fx, \quad F = w^{kl}, \quad w = \exp \frac{2\pi i}{n}$$

Question: Given $x$ in the QTT-format, can we compute $y$?

# FFT matrix and tensor networks

Let us do quantization again:

$$k = k_1 + 2k_2 + \ldots + 2^{d-1} k_d$$

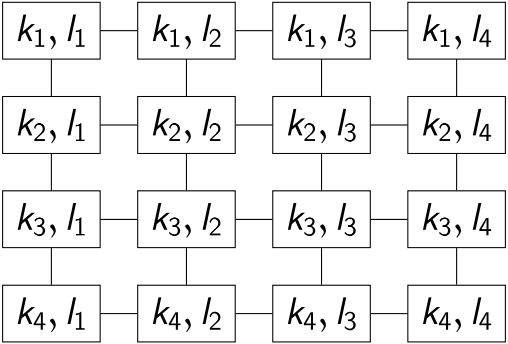
$$l = l_1 + 2l_2 + \ldots + 2^{d-1} l_d$$

$$w^{kl} = \prod_{pq} G_{pq}(k_p, l_q)$$

$$G_{pq}(k_p, l_q) = w^{2^{p-1} k_p 2^{q-1} l_q}$$

Rank-1 two-dimensional tensor network!

# FFT matrix and tensor networks(2)



# FFT of a vector

$$\sum_{l_1,\dots,l_d} \Big( \prod_{pq} G_{pq}(k_p, l_q) \Big) x_1(l_1) \dots x_d(l_d)$$

It can be rewritten as a Hadamard product of $d$ rank-2 tensors!

Hint: $l_k$ takes only two values

Complexity: $\mathcal{O}(d^2 r^3 \log n)$

# Convolution

The convolution operation is defined as

$$c_i = \sum_j a_{i-j} b_j.$$

How to do it in the QTT-format?

# Convolution

Idea:

Write convolution as

$$c_i = \sum_{jk} E_{ijk} a_k b_j$$

$$E_{ijk} = \delta_{k-i+j}.$$

Write down the tensor $E$:

$$E_{ijk} = E(i_1, \ldots, i_d, j_1, \ldots, j_d, k_1, \ldots, k_{d+1}).$$

Permute dimensions:

$$(i_1, j_1, k_1, i_2, j_2, k_2, \ldots).$$

Find an explicit representation

# Convolution: results

Summary of the results (Hackbusch and Kazeev, Khoromskij, Tyrtyshnikov)

- Toeplitz matrix generated by QTT-vectors of rank $r$ has rank $2r$
- Convolution of two vectors of rank $r$ has rank $2r$
- Multidimensional Toeplitz matrices have the same rank bound, but two QTT-matvecs required

# Solving eigenvalue problems and linear systems

Now, let us go to more advanced problems:

- $Ax = \lambda x$
- $Ax = f$,

with $x = X(j_1, \ldots, j_d), \quad f = F(i_1, \ldots, i_d)$,

$$A(i_1, \ldots, i_d; j_1, \ldots, j_d)$$

# Tensor-structured solvers

- Path 1: Do iterative methods with truncation (Krylov, preconditioning, multigrid, etc.)
- Path 2: Use the information about the structure of the solution

# Using the information about the solution

How to use the information about the solution?

# Using the information about the solution

How to use the information about the solution?

Formulate as an optimization problem!

# Formulation as an optimization problem

$$Ax = \lambda x, \ A = A^*,$$

We can minimize the Rayleigh quotient:

$$\frac{(Ax, x)}{(x, x)} \to \min,$$

Minimize not over the whole space,

but over the set of structured tensors!

- Nonconvex optimization problem
- Have to guess the ranks

# Idea of alternating iterations

The idea of alternating iterations is simple:

- Fix all except $X_k(i_k)$.
- The local problem reduces to the linear eigenvalue problem
- Guess the rank!

# Idea of DMRG

Here comes the wonderful idea of DMRG (Density Matrix Renormalization Group, S. White, 1993)

Generalization of the Wilson renormalization group (=ALS)

Optimize not over one core, but over a pair of cores, $X_k$ and $X_{k+1}$ !

# Idea of DMRG

$$X(i_1, i_2, i_3, i_4) = X_1(i_1)X_2(i_2)X_3(i_3)X_4(i_4) =$$
$$= W_{12}(i_1, i_2)X_3(i_3)X_4(i_4).$$

- Solve for $W_{12}$
- Split back by the SVD:
  $W_{12}(i_1, i_2) = X_1(i_1)X_2(i_2)$.

The rank is determined adaptively!

# DMRG and QTT

The DMRG method creates modes of size $n^2$.

- Very good for spin systems ($n = 2$ or $n = 4$)
- Very good for the QTT ($n = 2$).

# DMRG and QTT: papers

- Holtz, S., Rohwedder, T., Schneider, R., The alternating linear scheme for tensor optimization in the tensor train format (idea)
- S.V. Dolgov, I.V. Oseledets, Solution of linear systems and matrix inversion in the TT-format (working code)

# Difficulties in the DMRG

S.V. Dolgov, I.V. Oseledets Solution of linear systems and matrix inversion in the TT-format

- SVD-based truncation: $L_2$-norm approximation of $x$, but the equation can be differential
- How to avoid local minima
- Fast solution of local systems
- Appplication to matrix inversion

# Large mode sizes

What if the mode size is large?

Basically, the question is now how to increase the $k$-th rank.

Answer: Using (projected) residuals of the Krylov methods!

# Large mode sizes

The $k$-th ALS-step: The $x$ is in the special linear subspace: $x = (U \otimes I \otimes V)\phi$,

Analogously to the rounding procedure $U$ and $V$ are structured-orthogonal matrix and $\phi$ is $r_{k-1}n_k r_k$.

The local problem is then

$$\widehat{U}^\top A \widehat{U} \phi = \widehat{U} f.$$

Can enrich the basis with the (low-rank) approximation of the two-block residual

# AMEN: further details

S.V. Dolgov and D.V. Savostyanov, 2013:

- Alternating minimal energy methods for linear systems in higher dimensions. Part I/II

For more details: convergence estimates, algorithmic details and so on.

Still A LOT to be done on algorithms...

# Solving linear systems and eigenvalue problems

(Demo)

Solving the high-dimensional Poisson equation in the QTT-format:

$$\Delta u = f.$$

# Other important problems

- Solving block eigenvalue problems (minimizing block Rayleigh quotient)
- Solving nonstationary problems $\frac{dy}{dt} = Ay$, how to rewrite as a minimization problem

# Next lecture

The plan for the next (and the last!)

- Applications, new results, open problems