Low-Rank Tensor Techniques
for High-Dimensional Problems

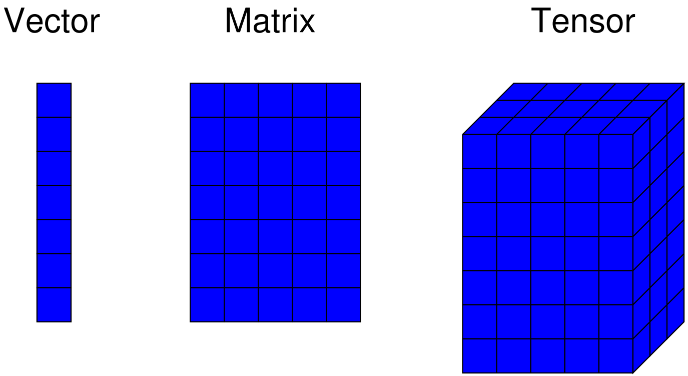
Daniel Kressner
CADMOS Chair for Numerical Algorithms and HPC
MATHICSE, EPFL

Contents

What is a tensor?

- Vectors, matrices, and tensors
- Basic calculus with tensors
- Vectorization and matricization
- μ-mode matrix products
- Two classes of tensor problems

Vectors, matrices, and tensors



- scalar $=$ tensor of order 0
- (column) vector $=$ tensor of order 1
- matrix $=$ tensor of order 2
- tensor of order 3
  $= n_1 n_2 n_3$ numbers arranged in $n_1 \times n_2 \times n_3$ array

Tensors of arbitrary order

A d-th order tensor $\mathcal{X}$ of size $n_1 \times n_2 \times \cdots \times n_d$ is a d-dimensional array with entries

$$\mathcal{X}_{i_1,i_2,\ldots,i_d}, \qquad i_\mu \in \{1,\ldots,n_\mu\} \text{ for } \mu = 1,\ldots,d.$$

In the following, entries of $\mathcal{X}$ are real (for simplicity) $\rightsquigarrow$

$$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}.$$

Multi-index notation:

$$\mathfrak{I} = \{1,\ldots,n_1\} \times \{1,\ldots,n_2\} \times \cdots \times \{1,\ldots,n_d\}.$$

Then $i \in \mathfrak{I}$ is a tuple of d indices:

$$i = (i_1, i_2, \ldots, i_d).$$

Allows to write entries of $\mathcal{X}$ as $\mathcal{X}_i$ for $i \in \mathfrak{I}$.

Two important points

1. A matrix $A \in \mathbb{R}^{m \times n}$ has a natural interpretation as a linear operator in terms of matrix-vector multiplications:

$$A : \mathbb{R}^n \to \mathbb{R}^m, \quad A : x \mapsto A \cdot x.$$

   There is no such (unique and natural) interpretation for tensors!

   $\rightsquigarrow$ fundamental difficulty to define meaningful general notion of eigenvalues and singular values of tensors.

2. Number of entries in tensor grows exponentially with d $\rightsquigarrow$

   Curse of dimensionality.

   Example: Tensor of order 30 with $n_1 = n_2 = \cdots = n_d = 10$ has 10^{30} entries $= 8 \times 10^{12}$ Exabyte storage![1]

   For $d \gg 1$: Cannot afford to store tensor explicitly (in terms of its entries).

[1] Global data storage calculated at 295 exabyte, see http://www.bbc.co.uk/news/technology-12419672.

Basic calculus

- Addition of two equal-sized tensors $\mathcal{X}, \mathcal{Y}$:

$$\mathcal{Z} = \mathcal{X} + \mathcal{Y} \quad \Leftrightarrow \quad \mathcal{Z}_i = \mathcal{X}_i + \mathcal{Y}_i \quad \forall i \in \mathfrak{I}.$$

- Scalar product with $\alpha \in \mathbb{R}$:

$$\mathcal{Z} = \alpha\mathcal{X} \quad \Leftrightarrow \quad \mathcal{Z}_i = \alpha\mathcal{X}_i \quad \forall i \in \mathfrak{I}.$$

$\rightsquigarrow$ vector space structure.

- Inner product of two equal-sized tensors $\mathcal{X}, \mathcal{Y}$:

$$\langle \mathcal{X}, \mathcal{Y} \rangle := \sum_{i \in \mathfrak{I}} x_i y_i.$$

$\rightsquigarrow$ Induced norm

$$\|\mathcal{X}\| := \Big(\sum_{i \in \mathfrak{I}} x_i^2 \Big)^{1/2}$$

For a 2nd order tensor ($=$ matrix) this corresponds to the *Frobenius norm*.

Vectorization

Tensor $\mathcal{X}$ of size $n_1 \times n_2 \times \cdots \times n_d$ has $n_1 \cdot n_2 \cdots n_d$ entries
$\rightsquigarrow$ many ways to stack entries in a (loooong) column vector.

One possible choice:

The vectorization of $\mathcal{X}$ is denoted by $\mathrm{vec}(\mathcal{X})$, where

$$\mathrm{vec} : \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d} \to \mathbb{R}^{n_1 \cdot n_2 \cdots n_d}$$

stacks the entries of a tensor in reverse lexicographical order into a long column vector.

Remark: For $d = 2$, this is the usual way how matrices are vectorized.

$$A = \left[\begin{array}{c|c} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{array}\right] \quad \Rightarrow \quad \mathrm{vec}(A) = \left[\begin{array}{c} a_{11} \\ a_{21} \\ a_{31} \\ \hline a_{12} \\ a_{22} \\ a_{32} \end{array}\right]$$

Vectorization

Example: $d = 3$, $n_1 = 3$, $n_2 = 2$, $n_3 = 3$.

$$\operatorname{vec}(\mathcal{X}) = \begin{bmatrix} x_{111} \\ x_{112} \\ x_{113} \\ x_{121} \\ \vdots \\ \vdots \\ x_{321} \\ x_{322} \\ x_{323} \end{bmatrix}$$

Matricization

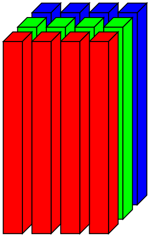
- A matrix has two modes (column mode and row mode).
- A dth-order tensor $\mathcal{X}$ has d modes ($\mu = 1$, $\mu = 2$, ..., $\mu = d$).

Let us fix all but one mode, e.g., $\mu = 1$: Then

$$\mathcal{X}(:, i_2, i_3, \ldots, i_d) \qquad \text{(abuse of MATLAB notation)}$$

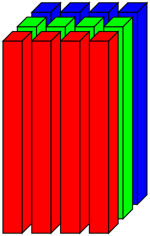
is a vector of length n_1 for each choice of $i_2, \ldots, i_d$.

$\rightsquigarrow$ View tensor $\mathcal{X}$ as a bunch of column vectors:

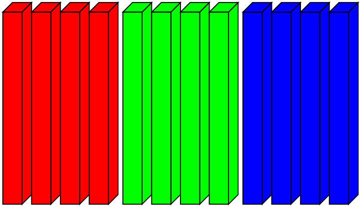


Matricization

Stack vectors into an $n_1 \times (n_2 \cdots n_d)$ matrix:



$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$



$X^{(1)} \in \mathbb{R}^{n_1 \times (n_2 n_3 \cdots n_d)}$

For $\mu = 1, \ldots, d$, the μ-mode matricization of $\mathcal{X}$ is a matrix

$$X^{(\mu)} \in \mathbb{R}^{n_\mu \times (n_1 \cdots n_{\mu-1} n_{\mu+1} \cdots n_d)}$$

with entries

$$\left(X^{(\mu)}\right)_{i_{\mu_1}, (i_1, \ldots, i_{\mu-1}, i_{\mu+1} \ldots i_d)} = \mathcal{X}_i \quad \forall i \in \mathfrak{I}.$$

Matricization

In MATLAB: `a = rand(2,3,4,5);`

- 1-mode matricization:

```
reshape(a,2,3*4*5)
```

- 2-mode matricization:

```
b = permute(a,[2 1 3 4]);
reshape(b,3,2*4*5)
```

- 3-mode matricization:

```
b = permute(a,[3 1 2 4]);
reshape(b,4,2*3*5)
```

- 4-mode matricization:

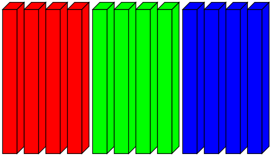
```
b = permute(a,[4 1 2 3]);
reshape(b,5,2*3*4)
```

For a matrix $A \in \mathbb{R}^{n_1 \times n_2}$:

$$A^{(1)} = A, \qquad A^{(2)} = A^T.$$

μ-mode matrix products

Consider 1-mode matricization $X^{(1)} \in \mathbb{R}^{n_1 \times (n_2 \cdots n_d)}$:



Seems to make sense to multiply an $m \times n_1$ matrix A from the left:

$$Y^{(1)} := A\,X^{(1)} \in \mathbb{R}^{m \times (n_2 \cdots n_d)}.$$

Can rearrange $Y^{(1)}$ back into an $m \times n_2 \times \cdots \times n_d$ tensor $\mathcal{Y}$.

This is called 1-mode matrix multiplication

$$\mathcal{Y} = A \circ_1 \mathcal{X} \qquad \Leftrightarrow \qquad Y^{(1)} = AX^{(1)}$$

More formally (and more ugly):

$$\mathcal{Y}_{i_1, i_2, \ldots, i_d} = \sum_{k=1}^{n_1} a_{i_1, k} \mathcal{X}_{k, i_2, \ldots, i_d}.$$

μ-mode matrix products

General definition of a μ-mode matrix product with $A \in \mathbb{R}^{m \times n_1}$:

$$\mathcal{Y} = A \circ_\mu \mathcal{X} \qquad \Leftrightarrow \qquad Y^{(\mu)} = AX^{(\mu)}.$$

More formally (and more ugly):

$$\mathcal{Y}_{i_1, i_2, \ldots, i_d} = \sum_{k=1}^{n_1} a_{i_\mu, k} \mathcal{X}_{i_1, \ldots, i_{\mu-1}, k, i_{\mu+1}, \ldots, i_d}.$$

For matrices:

- 1-mode multiplication $=$ multiplication from the left:

$$Y = A \circ_1 X = A X.$$

- 2-mode multiplication $=$ *transposed* multiplication from the right:

$$Y = A \circ_2 X = X A^T.$$

Kronecker product

For $m \times n$ matrix A and $k \times \ell$ matrix B, Kronecker product defined as

$$B \otimes A := \begin{bmatrix} b_{11}A & \cdots & b_{1\ell}A \\ \vdots & & \vdots \\ b_{k1}A & \cdots & b_{k\ell}A \end{bmatrix} \in \mathbb{R}^{km \times \ell n}.$$

Most important properties (for our purposes):

1. $\mathrm{vec}(A\,X) = (I \otimes A)\,\mathrm{vec}(X)$.
2. $\mathrm{vec}(X\,A^T) = (A \otimes I)\,\mathrm{vec}(X)$.
3. $(B \otimes A)(D \otimes C) = (BD \otimes AC)$.
4. $I_m \otimes I_n = I_{mn}$.

μ-mode matrix products and vectorization

By definition,

$$\mathrm{vec}(\mathcal{X}) = \mathrm{vec}\big(X^{(1)}\big).$$

Consequently, also

$$\mathrm{vec}(A \circ_1 \mathcal{X}) = \mathrm{vec}\big(A\, X^{(1)}\big).$$

$\rightsquigarrow$ Vectorized version of 1-mode matrix product:

$$\begin{aligned} \mathrm{vec}(A \circ_1 \mathcal{X}) &= (I_{n_2 \cdots n_d} \otimes A)\mathrm{vec}(\mathcal{X}) \\ &= (I_{n_d} \otimes \cdots \otimes I_{n_2} \otimes A)\, \mathrm{vec}(\mathcal{X}). \end{aligned}$$

Relation between μ-mode matrix product and matrix-vector product:

$$\mathrm{vec}(A \circ_\mu \mathcal{X}) = (I_{n_d} \otimes \cdots \otimes I_{n_{\mu+1}} \otimes A \otimes I_{n_{\mu-1}} \otimes \cdots \otimes I_{n_1})\, \mathrm{vec}(\mathcal{X})$$

Two classes of tensor problems

Class 1: function-related tensors

Consider a function $u(\xi_1, \ldots, \xi_d) \in \mathbb{R}$ in d variables $\xi_1, \ldots, \xi_d$.
Tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ represents discretization of u:

- $\mathcal{U}$ contains function values of u evaluated on a grid; **or**
- $\mathcal{U}$ contains coefficients of truncated expansion in tensorized basis functions:

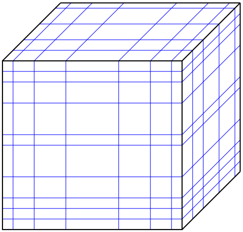
$$u(\xi_1, \ldots, \xi_d) \approx \sum_{i \in \mathfrak{I}} \mathcal{U}_i \, \phi_{i_1}(\xi_1)\phi_{i_2}(\xi_2) \cdots \phi_{i_d}(\xi_d).$$

Typical setting:

- $\mathcal{U}$ only given implicitly, e.g., as the solution of a discretized PDE;
- seek approximations to $\mathcal{U}$ with very low storage and tolerable accuracy.
- d may become very large.

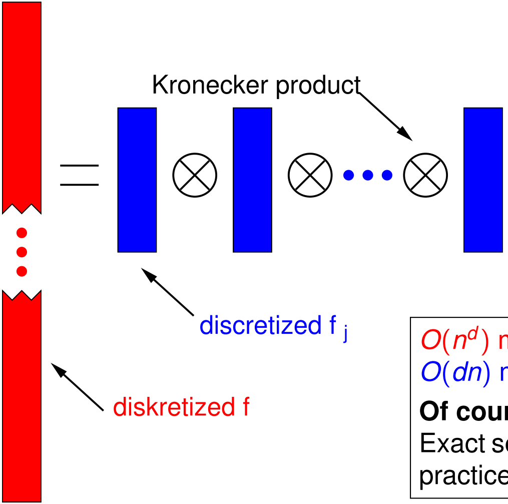
Focus of this lecture on function-related tensors!

Discretization of function in d variables
$\xi_1, \ldots, \xi_d \in [0, 1]$
$\rightsquigarrow$ #function values grows exponentially with d



Separability helps

Ideal situation:
Function f separable:
$f(\xi_1, \xi_2, \ldots, \xi_d) = f_1(\xi_1) f_2(\xi_2) \ldots f_d(\xi_d)$



$O(n^d)$ memory $\rightsquigarrow$
$O(dn)$ memory

Of course:
Exact separability rarely satisfied in practice.

Two classes of tensor problems

Class 2: data-related tensors

Tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ contains multi-dimensional data.

Example 1: $\mathcal{U}_{2011,3,2}$ denotes the number of papers published 2011 by author 3 in the mathematical journal 2.

Example 2: A video of 1000 frames with resolution 640×480 can be viewed as a $640 \times 480 \times 1000$ tensor.

Typical setting:

- entries of $\mathcal{U}$ given explicitly (at least partially).
- extraction of dominant features from $\mathcal{U}$.
- usually moderate values for d.

Summary

- Tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is a *d*-dimensional array.
- Various ways of reshaping entries of a tensor $\mathcal{X}$ into a vector or matrix.
- μ-mode matrix multiplication can be expressed with Kronecker products

Further reading:

- T. Kolda and B. W. Bader. Tensor decompositions and applications. SIAM Rev. 51 (2009), no. 3, 455–500.

Software:

- MATLAB offers basic functionality to work with *d*-dimensional arrays.
- MATLAB Tensor Toolbox: `http://www.csmr.ca.sandia.gov/~tgkolda/TensorToolbox/`

Applications in scientific computing

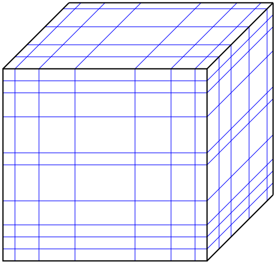
- High-dimensional elliptic PDEs
- High-dimensional PDE-eigenvalue problems
- Quantum many-body problems
- Stochastic Automata Networks
- further applications

High-dimensional elliptic PDEs: 3D model problem

- Consider

$$-\Delta u = f \quad \text{in } \Omega, \quad u|_{\partial\Omega} = 0,$$

on unit cube $\Omega = [0, 1]^3$.



- Discretize on tensor grid.
  Uniform grid for simplicity:

$$\xi_\mu^{(j)} = jh, \quad h = \frac{1}{n+1}$$

for $\mu = 1, 2, 3$.

- Approximate solution tensor $\mathcal{U} \in \mathbb{R}^{n \times n \times n}$:

$$\mathcal{U}_{i_1,i_2,i_3} \approx u\big(\xi_1^{(i_1)}, \xi_2^{(i_2)}, \dots, \xi_d^{(i_d)}\big).$$

High-dimensional elliptic PDEs: 3D model problem

- Discretization of 1D-Laplace:

$$-\partial_{xx} \approx \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} =: A.$$

- Application in each coordinate direction:

$$\begin{aligned} -\partial_{\xi_1\xi_1} u(\xi_1, \xi_2, \xi_3) &\approx A \circ_1 \mathcal{U}, \\ -\partial_{\xi_2\xi_2} u(\xi_1, \xi_2, \xi_3) &\approx A \circ_2 \mathcal{U}, \\ -\partial_{\xi_3\xi_3} u(\xi_1, \xi_2, \xi_3) &\approx A \circ_3 \mathcal{U}. \end{aligned}$$

- Hence,

$$-\Delta u \approx A \circ_1 \mathcal{U} + A \circ_2 \mathcal{U} + A \circ_3 \mathcal{U}$$

  or in vectorized form with $\mathbf{u} = \mathrm{vec}(\mathcal{U})$:

$$-\Delta u \approx (I \otimes I \otimes A + I \otimes A \otimes I + A \otimes I \otimes I)\mathbf{u}.$$

High-dimensional elliptic PDEs: 3D model problem

Finite difference discretization of model problem

$$-\Delta u = f \quad \text{in } \Omega, \quad u|_{\partial\Omega} = 0$$

for $\Omega = [0,1]^3$ takes the form

$$(I \otimes I \otimes A + I \otimes A \otimes I + A \otimes I \otimes I)\mathbf{u} = \mathbf{f}.$$

Similar structure for finite element discretization with tensorized FEs:

$$\mathbb{V} \otimes \mathbb{W} \otimes \mathbb{Z} = \Big\{ \sum \alpha_{ijk} v_i(\xi_1) w_j(\xi_2) z_k(\xi_3) : \ \alpha_{ijk} \in \mathbb{R} \Big\}$$

with

$$\mathbb{V} = \{v_1(\xi_1), \dots, v_n(\xi_1)\}, \ \mathbb{W} = \{w_1(\xi_2), \dots, w_n(\xi_2)\}, \ \mathbb{Z} = \{z_1(\xi_3), \dots, z_n(\xi_3)\}$$

Galerkin discretization

$$(K_V \otimes M_W \otimes M_Z + M_V \otimes K_W \otimes M_Z + M_V \otimes M_W \otimes K_Z)\mathbf{u} = \mathbf{f},$$

with 1D mass/stiffness matrices $M_V, M_W, M_Z, K_V, K_W, K_Z$.

High-dimensional elliptic PDEs: Arbitrary dimensions

Finite difference discretization of model problem

$$-\Delta u = f \quad \text{in } \Omega, \quad u|_{\partial\Omega} = 0$$

for $\Omega = [0, 1]^d$ takes the form

$$\Big(\sum_{j=1}^{d} I \otimes \cdots \otimes I \otimes A \otimes I \otimes \cdots \otimes I\Big)\mathbf{u} = \mathbf{f}.$$

To obtain such Kronecker structure in general:

- tensorized domain;
- highly structured grid;
- coefficients that can be written/approximated as sum of separable functions.

High-dimensional PDE-eigenvalue problems

PDE-eigenvalue problem

$$\begin{aligned} \Delta u(\xi) + V(\xi)u(\xi) &= \lambda\, u(\xi) && \text{in } \Omega = [0,1]^d, \\ u(\xi) &= 0 && \text{on } \partial\Omega. \end{aligned}$$

Assumption: Potential represented as

$$V(\xi) = \sum_{j=1}^{s} V_j^{(1)}(\xi_1)\, V_j^{(2)}(\xi_2) \cdots V_j^{(d)}(\xi_d).$$

$\rightsquigarrow$ finite difference discretization

$$\mathcal{A}\mathbf{u} = (\mathcal{A}_L + \mathcal{A}_V)\mathbf{u} = \lambda \mathbf{u},$$

with

$$\begin{aligned} \mathcal{A}_L &= \sum_{j=1}^{d} \underbrace{I \otimes \cdots \otimes I}_{d-j \text{ times}} \otimes A_L \otimes \underbrace{I \otimes \cdots \otimes I}_{j-1 \text{ times}}, \\ \mathcal{A}_V &= \sum_{j=1}^{s} A_{V,j}^{(d)} \otimes \cdots \otimes A_{V,j}^{(2)} \otimes A_{V,j}^{(1)}. \end{aligned}$$

Quantum many-body problems

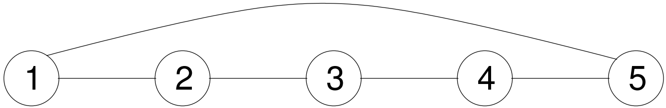
- spin-½ particles: proton, neutron, electron, and quark.
- two states: spin-up, spin-down
- quantum state for each spin represented by vector in $\mathbb{C}^2$ (spinor)
- quantum state for system of d spins represented by vector in $\mathbb{C}^{2^d}$
- quantum mechanical operators expressed in terms of Pauli matrices

$$P_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad P_y = \begin{bmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix}, \quad P_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

- spin Hamiltonian: sum of Kronecker products of Pauli matrices and identities
  $\rightsquigarrow$ each term describes physical (inter)action of spins
- interaction of spins described by graph
- Goal: Compute ground state of spin Hamiltonian.

Quantum many-body problems

Example: 1d chain of 5 spins with periodic boundary conditions



Hamiltonian describing pairwise interaction between nearest neighbors:

$$
\begin{aligned}
H \quad = & \quad P_z \otimes P_z \otimes I \otimes I \otimes I \\
+ & \quad I \otimes P_z \otimes P_z \otimes I \otimes I \\
+ & \quad I \otimes I \otimes P_z \otimes P_z \otimes I \\
+ & \quad I \otimes I \otimes I \otimes P_z \otimes P_z \\
+ & \quad P_z \otimes I \otimes I \otimes I \otimes P_z
\end{aligned}
$$

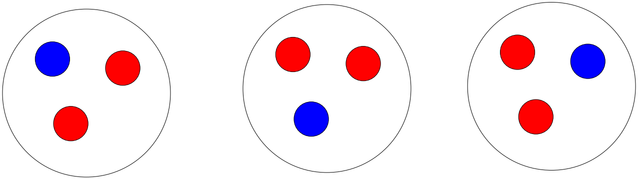
Quantum many-body problems

- Ising (ZZ) model for 1d chain of d spins with open boundary conditions:

$$
\begin{aligned}
H &= \sum_{k=1}^{p-1} I \otimes \cdots \otimes I \otimes P_z \otimes P_z \otimes I \otimes \cdots \otimes I \\
&\quad + \lambda \sum_{k=1}^{p} I \otimes \cdots \otimes I \otimes P_x \otimes I \otimes \cdots \otimes I
\end{aligned}
$$

  $\lambda =$ ratio between strength of magnetic field and pairwise interactions
- 1d Heisenberg (XY) model
- Current research: 2d models.
- More details in:
  Huckle/Waldherr/Schulte-Herbrüggen: Computations in Quantum Tensor Networks.
  Schollwöck: The density-matrix renormalization group in the age of matrix product states.

Stochastic Automata Networks (SANs)

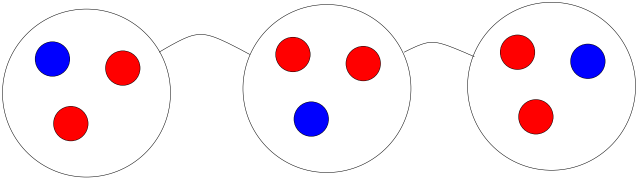


- 3 stochastic automata A_1, A_2, A_3 having 3 states each.
- Vector $x_t^{(i)} \in \mathbb{R}^3$ describes probabilities of states (1), (2), (3) in A_i at time t
- No coupling between automata $\rightsquigarrow$ local transition $x_t^{(i)} \mapsto x_{t+1}^{(i)}$ described by Markov chain:

$$x_{t+1}^{(i)} = E_i x_t^{(i)},$$

  with a stochastic matrix E_i.
- Stationary distribution of A_i = Perron vector of E_i (eigenvector for eigenvalue 1).

Stochastic Automata Networks (SANs)



- 3 stochastic automata A_1, A_2, A_3 having 3 states each.
- Coupling between automata $\rightsquigarrow$ local transition $x_t^{(i)} \mapsto x_{t+1}^{(i)}$ **not** described by Markov chain.
- Need to consider all possible combinations of states in (A_1, A_2, A_3):

$$(1,1,1),\ (1,1,2),\ (1,1,3),\ (1,2,1),\ (1,2,2),\ \ldots .$$

- Vector $x_t \in \mathbb{R}^{3^3}$ (or tensor $\mathcal{X}(t) \in \mathbb{R}^{3\times3\times3}$) describes probabilities of combined states.

Stochastic Automata Networks (SANs)

- Transition $x_t \mapsto x_{t+1}$ described by Markov chain:

$$x_{t+1} = \mathcal{E} x_t,$$

  with a large stochastic matrix $\mathcal{E}$.

- Oversimplified example:

$$\begin{aligned}
\mathcal{E} \quad &= \quad \underbrace{I \otimes I \otimes \widetilde{E}_1 + I \otimes \widetilde{E}_2 \otimes I + \widetilde{E}_3 \otimes I \otimes I}_{\text{local transition}}. \\
&+ \quad \underbrace{I \otimes E_{21} \otimes E_{12}}_{\text{interaction between } A_1, A_2} \quad + \quad \underbrace{E_{32} \otimes E_{23} \otimes I}_{\text{interaction between } A_2, A_3}
\end{aligned}$$

- Goal: Compute stationary distribution $=$ Perron vector of $\mathcal{E}$.
- More details in:
  Stewart: Introduction to the Numerical Solution of Markov Chains. Chapter 9.
  Buchholz: Product Form Approximations for Communicating Markov Processes.

Further applications

Other applications in scientific computing featuring low-rank tensor concepts:

- Boltzmann equation [Ibragimov/Rjasanow'2009].
- Dynamical systems [Koch/Lubich'2009].
- Parabolic PDEs [Andreev/Tobler'2011], [Khoromskij'2009].
- Stochastic PDEs [Khoromskij/Schwab'2010], [Matthies/Zander'2011], [Kressner/Tobler'2011], [Ballani/Grasedyck/Kluge'2011], ...
- Electronic structure calculation [Chinnamsetty et al.'2007], [Flad et al.'2009], [Khoromskij/Khoromskaja'2009], [Limpanuparb/Gill'2009], [Benedikt et al.'2011], [Mohlenkamp'2011], ...
- Evaluation of boundary integrals (in BEM): [Grasedyck], [Khoromskij/Sauter/Veit'2011].
- ...

Summary

- Large diversity of applications leading to linear systems / eigenvalue problems with Kronecker product structures.
- For many problems of practical interest:
  Explicit storage / computation of solution infeasible.
- Increasing use of low-rank tensor techniques.
  Heaviest use currently:
  DMRG for quantum many-body problems.
- Remark: For PDE-related applications, high dimensionality can also be addressed during the discretization phase (sparse grids, adaptive sparse discretization, ...).
  Has advantages and disadvantages.

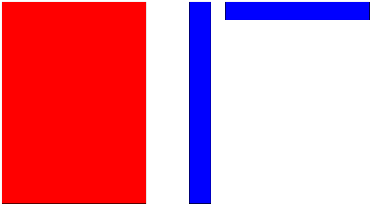
Approximate low-rank matrices

- Singular value decomposition
- Separability and low rank
- Separability by polynomial interpolation
- Separability by exponential sums
- Low rank of snapshot matrices

Low-rank approximation

Setting: Matrix $X \in \mathbb{R}^{n \times m}$, m and n too large to compute/store X explicitly.

Idea: Replace X by RS^T with $R \in \mathbb{R}^{n \times r}$, $S \in \mathbb{R}^{m \times r}$ and $r \ll m, n$.



	X	RS^T
Memory	nm	$nr + rm$
Cost	$\text{ops}(m, n)$	$\text{ops}(m, n) \times \frac{r}{\min\{m,n\}}$ (?)

$$\min\left\{\|X - RS^T\|_2 : \; R \in \mathbb{R}^{n \times r}, S \in \mathbb{R}^{m \times r}\right\} = \sigma_{k+1}.$$

with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}}$ of X.

Construction from singular value decomposition

SVD: Let matrix $X \in \mathbb{R}^{n\times m}$ and $k = \min\{m, n\}$. Then $\exists$ orthonormal matrices

$$U = \left[u_1,\, u_2,\, \ldots,\, u_k\right] \in \mathbb{R}^{n\times k}, \quad V = \left[v_1,\, v_2,\, \ldots,\, v_k\right] \in \mathbb{R}^{m\times k},$$

such that

$$X = U\Sigma V^T, \qquad \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k).$$

Choose $r \le k$ and partition

$$X = \left[U_1,\, U_2\right] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \left[V_1,\, V_2\right]^T = \underbrace{U_1\,\Sigma_1}_{=:R}\,\underbrace{V_1^T}_{=:S^T} + U_2\Sigma_2 V_2^T.$$

Then $\|X - RS^T\|_2 = \|\Sigma_2\|_2 = \sigma_{r+1}$.

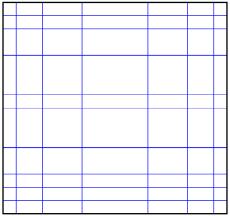
Good low rank approximation if singular values decay sufficiently fast.

Also: $\mathrm{span}(X) \approx \mathrm{span}(R)$, $\mathrm{span}(X^T) \approx \mathrm{span}(S^T)$

Discretization of bivariate function

- Bivariate function: $f(x, y) : [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \to \mathbb{R}$.
- Function values on tensor grid $[x_1, \dots, x_n] \times [y_1, \dots, y_m]$:

$$F = \begin{bmatrix} f(x_1, y_1) & f(x_1, y_2) & \cdots & f(x_1, y_n) \\ f(x_2, y_1) & f(x_2, y_2) & \cdots & f(x_2, y_n) \\ \vdots & \vdots & & \vdots \\ f(x_m, y_1) & f(x_m, y_2) & \cdots & f(x_m, y_n) \end{bmatrix}$$



Basic but crucial observation: $f(x, y) = g(x)h(y) \rightsquigarrow$

$$F = \begin{bmatrix} g(x_1)h(y_1) & \cdots & g(x_1)h(y_n) \\ \vdots & & \vdots \\ g(x_m)h(y_1) & \cdots & g(x_m)h(y_n) \end{bmatrix} = \begin{bmatrix} g(x_1) \\ \vdots \\ g(x_m) \end{bmatrix} \begin{bmatrix} h(y_1) & \cdots & h(y_n) \end{bmatrix}$$

Separability implies rank 1.

Separability and low rank

Approximation by sum of separable functions

$$f(x,y) = \underbrace{g_1(x)h_1(y) + \cdots + g_r(x)h_r(y)}_{=:f_r(x,y)} + \text{error}.$$

Define

$$F_r = \begin{bmatrix} f_r(x_1,y_1) & \cdots & f_r(x_1,y_n) \\ \vdots & & \vdots \\ f_r(x_m,y_1) & \cdots & f_r(x_m,y_n) \end{bmatrix}.$$

Then F_r has rank $\leq r$ and $\|F - F_r\|_F \leq \sqrt{mn} \times \text{error}$.

$\rightsquigarrow$

$$\sigma_{r+1}(F) \leq \|F - F_r\|_2 \leq \|F - F_r\|_F \leq \sqrt{mn} \times \text{error}.$$

Semi-separable approximation implies low-rank approximation.

Semi-separable approximation by polynomials

Solution of approximation problem

$$f(x, y) = g_1(x)h_1(y) + \cdots + g_r(x)h_r(y) + \text{error}.$$

not trivial; g_j, h_j can be chosen arbitrarily!

General construction by polynomial interpolation:

1. Lagrange interpolation of $f(x, y)$ in y-coordinate:

$$I_y[f](x, y) = \sum_{j=1}^{r} f(x, \theta_j) L_j(y)$$

   with Lagrange polynomials L_j of degree $r - 1$ on $[x_{\min}, x_{\max}]$.

2. Interpolation of $I_y[f]$ in x-coordinate:

$$I_x[I_y[f]](x, y) = \sum_{i,j=1}^{r} f(\xi_i, \theta_j) L_i(x) L_j(y) \mathrel{\hat{=}} \sum_{i=1}^{r} L_{i,x}(x) L_{j,y}(y),$$

   where $f[f(\xi_i, \theta_j)]_{i,j}$ is “diagonalized” by SVD.

Semi-separable approximation by polynomials

$$\begin{aligned}
\text{error} &\leq \|f - I_x[I_y[f]]\|_\infty \\
&= \|f - I_x[f] + I_x[f] - I_x[I_y[f]]\|_\infty \\
&\leq \|f - I_x[f]\|_\infty + \|I_x\|_\infty \|f - I_y[f]\|_\infty
\end{aligned}$$

with Lebesgue constant $\|I_x\|_\infty \sim \log r$ when using Chebyshev interpolation nodes.

Polynomial interpolation error typically much too pessimistic

- Lebesgue constants hit hard in high dimensions: $(\log r)^{d-1}$.
- Severe theoretical barriers for general smooth multivariate functions:
  E. Novak and H. Woźniakowski: Tractability of Multivariate Problems, Volume I and II. EMS.

Semi-separable approximation of $1/(x+y)$

Consider

$$f(x,y) = \frac{1}{x+y}, \qquad x,y \in [\alpha,\beta], \quad 0 < \alpha < \beta.$$

Apply numerical quadrature:

$$\frac{1}{z} = \int_0^\infty e^{-tz}\,\mathrm{d}t = \sum_{j=1}^{r} \omega_j e^{-\gamma_j z} + \text{error}.$$

Inserting $z = x + y \rightsquigarrow$

$$\frac{1}{x+y} = \sum_{j=1}^{r} \omega_j e^{-\gamma_j (x+y)} + \text{error} = \sum_{j=1}^{r} \omega_j e^{-\gamma_j x} e^{-\gamma_j y} + \text{error}.$$

Choice of nodes $\gamma_j > 0$ and weights $\omega_j > 0$ as in [Stenger'93, Braess'86, Braess/Hackbusch'05] $\rightsquigarrow$

$$\text{error} \le \frac{8}{|\alpha|} \exp\left[-\frac{r\pi^2}{\log(8\beta/\alpha)}\right].$$

Semi-separable approximation by exponential sums

- Consider more general case of function $f(x,y) := g(x+y)$.
- Approximation of $g(z)$ with $z := x + y$ by exponential sum

$$g(z) \approx \sum_{j=1}^{r} \omega_j \exp(\gamma_j z) \tag{1}$$

  for some coefficients $\gamma_j, \omega_j \in \mathbb{R}$.
- (1) gives semi-separable approximation for f:

$$\begin{aligned} f(x,y) = g(x+y) &\approx \sum_{j=1}^{r} \omega_j \exp(\gamma_j(x+y)) \\ &= \sum_{j=1}^{r} \omega_j \exp(\gamma_j x) \exp(\gamma_j y). \end{aligned}$$

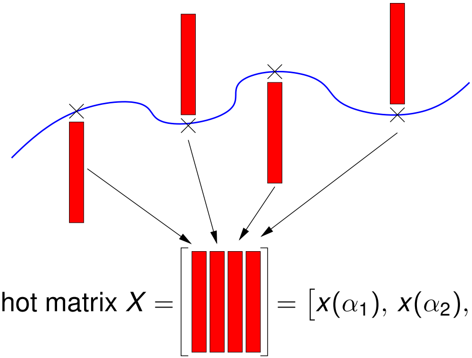
- Naturally extends to arbitrarily many variables.
- Problem: (1) nontrivial approx problem [Braess'1986], [Hackbusch'2006], ...

Low-rank approximation of snapshot matrices

Vector-valued function

$$x(\alpha) : [\alpha_{\min}, \alpha_{\max}] \to \mathbb{R}^n$$

Sampling at $\alpha_1, \dots, \alpha_m \in [\alpha_{\min}, \alpha_{\max}]$:



Snapshot matrix $X = \Big[\ \Big] = \big[x(\alpha_1),\ x(\alpha_2),\ \dots,\ x(\alpha_m)\big]$

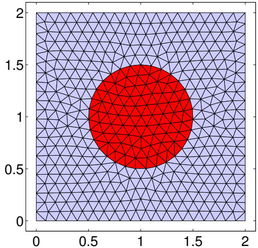
Example: Baking 1 cookie

Stationary heat equation with pw constant heat conductivity $\sigma(x, \alpha)$:

$$\begin{aligned} -\nabla(\sigma(x,\alpha)\nabla u) &= f \quad \text{in } \Omega = [-1,1]^2 \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

- $\sigma(\text{baking tray}) = 1$
- $\sigma(\text{cookie}) = 1 + \alpha$
- Undetermined parameter

$$\alpha \in [\alpha_{\text{min}}, \alpha_{\text{max}}].$$



Vertices : 455, # Elements : 825, # Edges : 1279

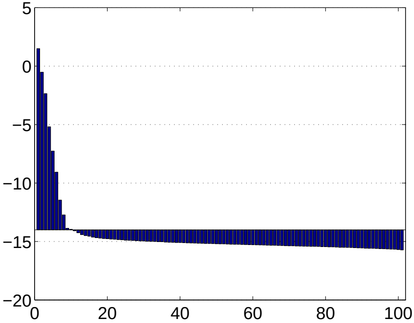
Standard FE discretization results in *linearly* parameter-dependent linear system

$$(A_0 + \alpha A_1)x(\alpha) = b.$$

Singular value decay – observation

- 1 Cookie: $n = 371, m = 101$.

log$_{10}$(singular values of snapshot matrix)



- Foundation of Proper Orthogonal Decomposition and Reduced Basis Methods.

Singular value decay – explanation

Polynomial approximation:

$$x(\alpha) = x_0 + \alpha x_1 + \alpha^2 x_2 + \cdots + \alpha^{k-1} x_{k-1} + \text{error}.$$

Approximation error:

- Assume $b(\cdot)$, $A(\cdot)$ analytic $\rightsquigarrow$ $x(\cdot)$ analytic.
- Then
$$\text{error} \lesssim \rho^{-k},$$
where $\rho > 1$ depends on domain of analyticity of A, b.
(Proof: Direct extension of classical result for scalar-valued functions.)

Singular value decay – explanation

Polynomial approximation:

$$x(\alpha) = x_0 + \alpha x_1 + \alpha^2 x_2 + \cdots + \alpha^{k-1} x_{k-1} + \text{error}.$$

Snapshot matrix:

$$\begin{aligned} X &= \big[x(\alpha_1),\, x(\alpha_2),\, \ldots,\, x(\alpha_m)\big] \\ &= \big[x_0,\, x_1,\, \ldots,\, x_{k-1}\big] \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_m \\ \vdots & \vdots & & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \ldots & \alpha_m^{k-1} \end{bmatrix} + \text{error} \\ &= \text{matrix of rank } k + \text{error} \end{aligned}$$

$$\sigma_{k+1}(X) \leq \text{error} \lesssim \rho^{-k}$$

Remark: Trivially extends to pw analytic case.

Singular value decay – pw analytic case

Example: Consider smallest singular value $\sigma(z)$ and corresponding right singular vector $v(z)$ of $B(z) = A - \mathrm{i}zI$ for $z \in [-1, 1]$.

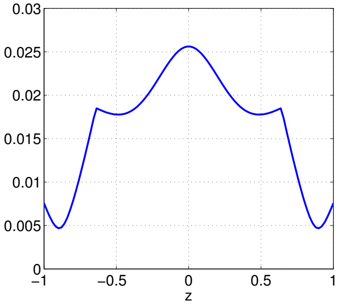
- $s(z)$ only Lipschitz cont, but pw anal.
- $v(z)$ discontinuous, but pw anal.

- $A = 2 \times 2$ block diag `randn`, $n = 400$.
- Snapshot matrix of singular vectors:

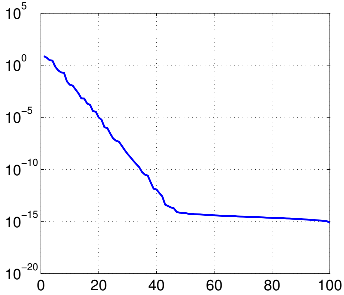
$$X = \left[\, v(z_1),\ v(z_2),\ \dots,\ v(z_{100}) \right]$$

for equidistant samples $z_j \in [-1, 1]$.

$\sigma(z)$



Singular values of X



Summary

Need strong singular value decay for good low-rank approximations.

For function-related matrices/tensors: Strong link to semi-separable approximations.

Smoothness seems to be important... at least somehow.

- Fortunately, smoothness is not necessary.
  Piecewise smoothness can be enough.
- Unfortunately, smoothness is not sufficient for higher-order tensors.
- Need to impose stronger regularity as dimension/order d increases, based, e.g., on mixed weak derivatives [Yserentant: Regularity and approximability of electronic wave functions. 2010].

Low-rank tensors:
CP and Tucker

- CP
- Tucker
- Higher-order SVD
- Tensor networks

CP decomposition

- Aim: Generalize concept of low rank from matrices to tensors.
- One possibility motivated by

$$
\begin{aligned}
X &= \big[a_1,\ a_2,\ \ldots,\ a_R\big]\big[b_1,\ b_2,\ \ldots,\ b_R\big]^T = \\
&= a_1 b_1^T + a_2 b_2^T + \cdots + a_R b_R^T.
\end{aligned}
$$

$\rightsquigarrow$ vectorization

$$
\mathrm{vec}(X) = b_1 \otimes a_1 + b_2 \otimes a_2 + \cdots + b_R \otimes a_R.
$$

Canonical **P**olyadic decomposition of tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ defined via

$$
\mathrm{vec}(\mathcal{X}) = c_1 \otimes b_1 \otimes a_1 + c_2 \otimes b_2 \otimes a_2 + \cdots + c_R \otimes b_R \otimes a_R
$$

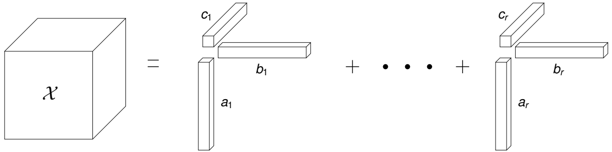
for vectors $a_j \in \mathbb{R}^{n_1}$, $b_j \in \mathbb{R}^{n_2}$, $c_j \in \mathbb{R}^{n_3}$.

CP directly corresponds to semi-separable approximation.
Tensor rank of $\mathcal{X}$ = minimal possible R

CP decomposition

Illustration of CP decomposition

$$\mathrm{vec}(\mathcal{X}) = c_1 \otimes b_1 \otimes a_1 + c_2 \otimes b_2 \otimes a_2 + \cdots + c_R \otimes b_R \otimes a_R.$$



CP decomposition

- CP decomposition offers low data-complexity; for constant R:

  linear complexity in d.

- For **matrices**:
  - rank r is upper semi-continuous $\rightsquigarrow$ closedness property:

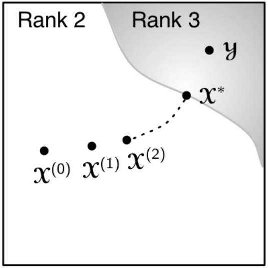
    sequence of rank$=r$ matrices can only converge to rank$\leq r$ matrix.
  - best low-rank approximation possible by successive rank-1 approximations.
  - Robust black-box algorithms/software available (`svd`, Lanczos).

For tensors of order $d \geq 3$:

- tensor rank R is **not** upper semi-continuous $\rightsquigarrow$

  lack of closedness

- successive rank-1 approximations fail
- all algorithms based on optimization techniques (ALS, Gauss-Newton)



Picture taken from [Kolda/Bader'2009].

Tucker decomposition

- Aim: Generalize concept of low rank from matrices to tensors.
- Alternative possibility motivated by

$$A = U \cdot \Sigma \cdot V^T, \qquad U \in \mathbb{R}^{n_1 \times r}, \quad V \in \mathbb{R}^{n_2 \times r}, \quad \Sigma \in \mathbb{R}^{r \times r}.$$

$\rightsquigarrow$ vectorization

$$\mathrm{vec}(X) = \big(V \otimes U\big) \cdot \mathrm{vec}(\Sigma).$$

Ignore diagonal structure of Σ and call it C.

Tucker decomposition of tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ defined via

$$\mathrm{vec}(\mathcal{X}) = \big(W \otimes V \otimes U\big) \cdot \mathrm{vec}(\mathcal{C})$$

with $U \in \mathbb{R}^{n_1 \times r_1}$, $V \in \mathbb{R}^{n_2 \times r_2}$, $W \in \mathbb{R}^{n_3 \times r_3}$,
and core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$.

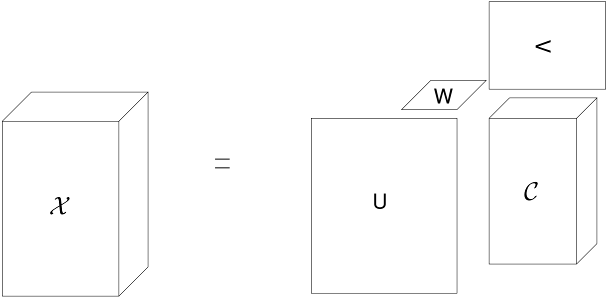
In terms of μ-mode matrix products:

$$\mathcal{X} = U \circ_1 V \circ_2 W \circ_3 \mathcal{C} =: (U, V, W) \circ \mathcal{C}.$$

Tucker decomposition

Illustration of Tucker decomposition

$$\mathcal{X} = (U, V, W) \circ \mathcal{C}$$



Tucker decomposition

Consider all three matricizations:

$$\begin{aligned} X^{(1)} &= U \cdot C^{(1)} \cdot \big(W \otimes V\big)^T, \\ X^{(2)} &= V \cdot C^{(2)} \cdot \big(W \otimes U\big)^T, \\ X^{(3)} &= W \cdot C^{(3)} \cdot \big(V \otimes U\big)^T. \end{aligned}$$

These are low rank decompositions $\rightsquigarrow$

$$\operatorname{rank}\big(X^{(1)}\big) \le r_1, \quad \operatorname{rank}\big(X^{(2)}\big) \le r_2, \quad \operatorname{rank}\big(X^{(3)}\big) \le r_3.$$

Multilinear rank of tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ defined by tuple

$$(r_1, r_2, r_3), \quad \text{with} \quad r_i = \operatorname{rank}\big(X^{(i)}\big).$$

Higher-order SVD (HOSVD)

Goal: Approximate given tensor $\mathcal{X}$ by Tucker decomposition with prescribed multilinear rank (r_1, r_2, r_3).

1. Calculate SVD of matricizations:

$$X^{(\mu)} = \widetilde{U}_\mu \widetilde{\Sigma}_\mu \widetilde{V}_\mu^T \quad \text{for } \mu = 1, 2, 3.$$

2. Truncate basis matrices:

$$U_\mu := \widetilde{U}_\mu(:, 1 : r_\mu) \quad \text{for } \mu = 1, 2, 3.$$

3. Form core tensor:

$$\text{vec}(\mathcal{C}) := \left(U_3^T \otimes U_2^T \otimes U_1^T\right) \cdot \text{vec}(\mathcal{X}).$$

Truncated tensor produced by HOSVD [Lathauwer/De Moor/Vandewalle'2000]:

$$\text{vec}(\widetilde{\mathcal{X}}) := \left(U_3 \otimes U_2 \otimes U_1\right) \cdot \text{vec}(\mathcal{C}).$$

Remark:
Orthogonal projection $\widetilde{\mathcal{X}} := \left(\pi_1 \circ \pi_2 \circ \pi_3\right)\mathcal{X}$ with $\pi_\mu \mathcal{X} := U_\mu U_\mu^T \circ_\mu \mathcal{X}$.

Higher-order SVD (HOSVD)

Tensor $\widetilde{\mathcal{X}}$ resulting from HOSVD satisfies quasi-optimality condition

$$\|\mathcal{X} - \widetilde{\mathcal{X}}\| \leq \sqrt{d}\|\mathcal{X} - \mathcal{X}_{\text{best}}\|,$$

where $\mathcal{X}_{\text{best}}$ is best approximation of $\mathcal{X}$ with multilinear ranks $(r_1, \ldots, r_d)$.

Proof:

$$\begin{aligned}
\|\mathcal{X} - \widetilde{\mathcal{X}}\|^2 &= \|\mathcal{X} - (\pi_1 \circ \pi_2 \circ \pi_3)\mathcal{X}\|^2 \\
&= \|\mathcal{X} - \pi_1\mathcal{X}\|^2 + \|\pi_1\mathcal{X} - (\pi_1 \circ \pi_2)\mathcal{X}\|^2 + \cdots \\
&\quad \cdots + \|(\pi_1 \circ \pi_2)\mathcal{X} - (\pi_1 \circ \pi_2 \circ \pi_3)\mathcal{X}\|^2 \\
&\leq \|\mathcal{X} - \pi_1\mathcal{X}\|^2 + \|\mathcal{X} - \pi_2\mathcal{X}\|^2 + \|\mathcal{X} - \pi_3\mathcal{X}\|^2
\end{aligned}$$

Using

$$\|\mathcal{X} - \pi_\mu\mathcal{X}\| \leq \|\mathcal{X} - \mathcal{X}_{\text{best}}\| \quad \text{for } \mu = 1, 2, 3$$

leads to

$$\|\mathcal{X} - \widetilde{\mathcal{X}}\|^2 \leq 3 \cdot \|\mathcal{X} - \mathcal{X}_{\text{best}}\|^2.$$

Best approximation: See [Kolda/Bader'09].

Tucker decomposition – Summary

For general tensors:

- multilinear rank r is upper semi-continuous $\rightsquigarrow$ closedness property.
- HOSVD – simple and robust algorithm to obtain quasi-optimal low-rank approximation.
- quasi-optimality good enough for most applications in scientific computing.
- robust black-box algorithms/software available (e.g., Tensor Toolbox).

Drawback:

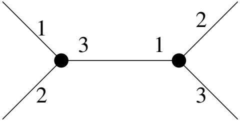
Storage of core tensor $\sim r^d$
$\rightsquigarrow$ curse of dimensionality

Tensor network diagrams

Tensor network = undirected graph with:

- each node is a tensor;
- each outgoing edge is a mode;
- each connected edge represents a contraction; example:

$$\mathcal{Z}_{i_1,i_2,i_3,i_4} = \sum_{j=1}^{r} \mathcal{X}_{i_1,i_2,j} \mathcal{Y}_{j,i_3,i_4}.$$



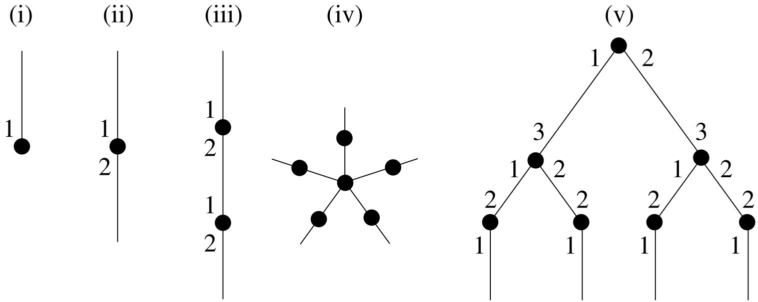
- number of free edges = order of tensor represented by entire network

Researchers on quantum many-body problems think[2] in terms of tensor networks!

[2] and dream

Tensor network diagrams

Examples:



(i) vector;
(ii) matrix;
(iii) matrix-matrix multiplication;
(iv) Tucker decomposition;
(v) hierarchical Tucker decomposition.

Low-rank tensors: Hierarchical Tucker

- Intro of Hierarchical Tucker Decomposition (HTD)
- MATLAB toolbox `htucker`
- Basic operations: μ-mode matrix multiplication, addition, ...
- Advanced Operations: inner product, elementwise multiplication, ...

Introduction

- CP offers low data complexity but difficult truncation;
- Tucker offers simple truncation but high data complexity.

Recently developed formats:

- Matrix Product State (MPS),
- TT decomposition,
- Hierarchical Tucker decomposition (HTD).

Aim to offer compromise between CP and Tucker.

Focus in this lecture: HTD.

- L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM J. Matrix Anal. Appl.*, 31(4):2029–2054, 2010.
- W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *J. Fourier Anal. Appl.*, 15(5):706–722, 2009.
- D. Kressner and C. Tobler. `htucker` – A MATLAB toolbox for the hierarchical Tucker decomposition. In preparation. See `http://www.math.ethz.ch/~ctobler`.

More general matricizations

Recall: μ-mode matricization for tensor $\mathcal{X}$,

$$X^{(\mu)} \in \mathbb{R}^{n_\mu \times (n_1 \cdots n_{\mu-1} n_{\mu+1} \cdots n_d)}, \qquad \mu = 1, \ldots, d.$$

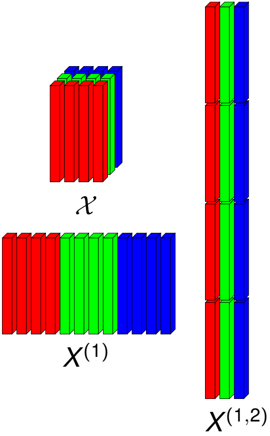
It is getting ugly...

General matricization for mode decomposition $\{1, \ldots, d\} = t \cup s$:

$$X^{(t)} \in \mathbb{R}^{(n_{t_1} \cdots n_{t_k}) \times (n_{s_1} \cdots n_{s_{d-k}})}$$

with

$$\left(X^{(t)}\right)_{(i_{t_1}, \ldots, i_{t_k}), (i_{s_1}, \ldots, i_{s_{d-k}})} := \mathcal{X}_{i_1, \ldots, i_d}.$$



Hierarchical construction

Singular value decomposition: $X^{(t)} = U_t \Sigma_t U_s^T$.

Column spaces are nested $\rightsquigarrow$

$$\begin{aligned} t = t_1 \cup t_2 \quad &\Rightarrow \quad \mathrm{span}(U_t) \subset \mathrm{span}(U_{t_2} \otimes U_{t_1}) \\ &\Rightarrow \quad \exists B_t : \; U_t = (U_{t_2} \otimes U_{t_1}) B_t. \end{aligned}$$

Size of U_t:

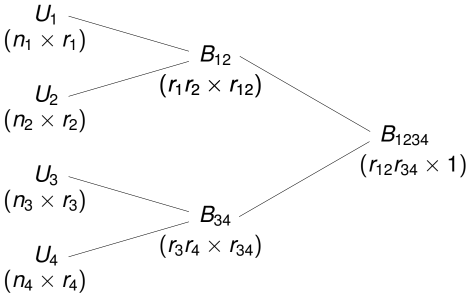
$$U_t \in \mathbb{R}^{n_{t_1} \cdots n_{t_k} \times r_t} \quad \text{with} \quad r_t = \mathrm{rank}\big(X^{(t)}\big).$$

For $d = 4$:

$$\begin{aligned} U_{12} &= (U_2 \otimes U_1) B_{12} \\ U_{34} &= (U_4 \otimes U_3) B_{34} \\ \mathrm{vec}(\mathcal{X}) = X^{(1234)} &= (U_{34} \otimes U_{12}) B_{1234} \\ \Rightarrow \quad \mathrm{vec}(\mathcal{X}) &= (U_4 \otimes U_3 \otimes U_2 \otimes U_1)(B_{34} \otimes B_{12}) B_{1234}. \end{aligned}$$

Dimension tree

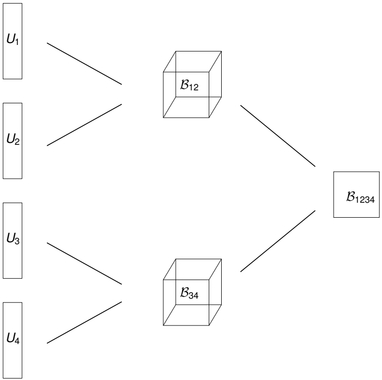
Tree structure for $d = 4$:



Reshape:

$$
\begin{aligned}
B_{12} \in \mathbb{R}^{r_1 r_2 \times r_{12}} \quad &\Rightarrow \quad \mathcal{B}_{12} \in \mathbb{R}^{r_1 \times r_2 \times r_{12}} \\
B_{34} \in \mathbb{R}^{r_3 r_4 \times r_{34}} \quad &\Rightarrow \quad \mathcal{B}_{34} \in \mathbb{R}^{r_3 \times r_4 \times r_{34}} \\
B_{1234} \in \mathbb{R}^{r_{12} r_{34} \times 1} \quad &\Rightarrow \quad \mathcal{B}_{1234} \in \mathbb{R}^{r_{12} \times r_{34}}
\end{aligned}
$$

Dimension tree



- Often, U_1, U_2, U_3, U_4 are orthonormal. This is advantageous but not required.
- Storage requirements for general d:

$$\mathcal{O}(dnr) + \mathcal{O}(dr^3),$$

where $r = max\{r_t\}$, $n = \max\{n_\mu\}$.

Constructors for MATLAB class `htensor`

`x = htensor([4 5 6 7])` constructs zero `htensor` of size $4 \times 5 \times 6 \times 7$, with a balanced dimension tree.

`x = htensor([4 5 6 7], 'TT')` constructs zero `htensor` of size $4 \times 5 \times 6 \times 7$, with a TT-style dimension tree.

`x = htensor({U1, U2, U3})` constructs `htensor` from tensor in CP decomp $\mathcal{X}(i_1, i_2, i_3) = \sum_j U_1(i_1, j) U_2(i_2, j) U_3(i_3, j)$.

`x = htenrandn([4 5 6 7])` constructs `htensor` of size $4 \times 5 \times 6 \times 7$, with random ranks and random entries.

`x = htenones([4 5 6 7])` constructs `htensor` of size $4 \times 5 \times 6 \times 7$, with all entries one.

...

Basic functionality for MATLAB class `htensor`

Example: `x` is in `htensor` of order 4.

`x(1, 3, 4, 2)` returns entry of $\mathcal{X}$.
`x(1, 3, :, :)` returns slice of $\mathcal{X}$ as an `htensor`.
`full(x)` returns full tensor represented by $\mathcal{X}$. (use with care)
`disp_tree(htenrand([5 4 6 3]))` returns tree structure/ranks:

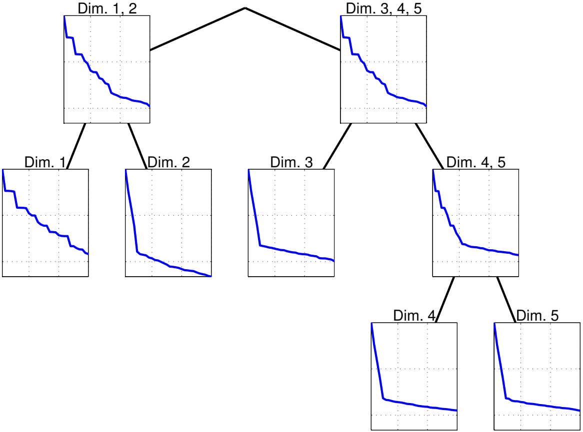
```
ans is an htensor of size 5 x 4 x 6 x 3
  1-4     1; 6 3 1
  1-2     2; 3 4 6
    1     4; 5 3
    2     5; 4 4
  3-4     3; 3 3 3
    3     6; 6 3
    4     7; 3 3
```

`spy(x)` displays spy plots of U_t, B_t, on the dimension tree.
`change_root(x, i)` switches root node.

Singular value tree

`plot_sv(x)` plots singular values of corresponding matricizations in the dimension tree of a tensor $\mathcal{X}$.

Example: Singular value tree of solution to elliptic PDE with 4 parameters.



Remark: Singular values are computed from Gramians.

Basic ops: μ-mode matrix multiplication

Application of matrix $A \in \mathbb{R}^{m \times n_\mu}$ to mode μ of $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$:

$$\mathcal{Y} = A \circ_\mu \mathcal{X} \quad \Leftrightarrow \quad Y^{(\mu)} = AX^{(\mu)}.$$

Nearly trivial if $\mathcal{X}$ is in $\mathcal{H}$-Tucker format:

$$\begin{aligned} A \circ_\mu \mathcal{X} &= A \circ_\mu \big((U_1, \dots, U_d) \circ \mathcal{C}\big) \\ &= (U_1, \dots, U_{\mu-1}, AU_\mu, U_{\mu+1}, \dots, U_d) \circ \mathcal{C} \end{aligned}$$

- Almost no operations required.
- Ranks stay the same.
- Orthogonality destroyed.

`ttm(x, A, 2)` applies matrix A to `htensor` $\mathcal{X}$ in mode 2.
`y = ttm(x, {A, B, C}, [2, 3, 4])`
`y = ttm(x, @(x)(fft(x)), 2)` applies FFT in mode 2.
`y = ttm(x, {A, B, C}, [2, 3, 4], 'h')` successively applies matrices A^T, B^T, C^T in modes $2, 3, 4$.

Addition of low-rank matrices

Addition of two matrices in low-rank format:

$$A = U_1 \Sigma_A U_2^T, \qquad B = V_1 \Sigma_B V_2^T$$

$\Rightarrow$

$$A + B = \begin{bmatrix} U_1 & V_1 \end{bmatrix} \begin{bmatrix} \Sigma_A & 0 \\ 0 & \Sigma_B \end{bmatrix} \begin{bmatrix} U_2 & V_2 \end{bmatrix}^T$$

- No operations required.
- Rank increases.
- Orthogonality destroyed.

Addition of low-rank tensors

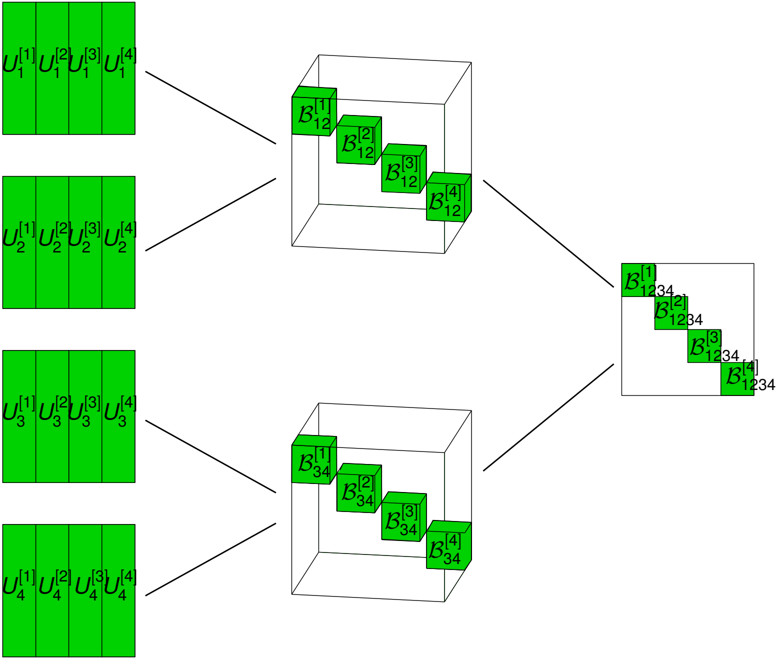
Addition of four tensors $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4$ in $\mathcal{H}$-Tucker format:

$$\mathcal{X}_1 + \mathcal{X}_2 + \mathcal{X}_3 + \mathcal{X}_4.$$

Proceed as in matrix case by embedding factors in larger matrices.

- No operations required.
- $\mathcal{H}$-Tucker rank increases.
- Orthogonality destroyed.

Command in `htucker`: `x1 + x2 + x3 + x4`


$U_1^{[1]}$ $U_1^{[2]}$ $U_1^{[3]}$ $U_1^{[4]}$
$U_2^{[1]}$ $U_2^{[2]}$ $U_2^{[3]}$ $U_2^{[4]}$
$U_3^{[1]}$ $U_3^{[2]}$ $U_3^{[3]}$ $U_3^{[4]}$
$U_4^{[1]}$ $U_4^{[2]}$ $U_4^{[3]}$ $U_4^{[4]}$
$\mathcal{B}_{12}^{[1]}$ $\mathcal{B}_{12}^{[2]}$ $\mathcal{B}_{12}^{[3]}$ $\mathcal{B}_{12}^{[4]}$
$\mathcal{B}_{34}^{[1]}$ $\mathcal{B}_{34}^{[2]}$ $\mathcal{B}_{34}^{[3]}$ $\mathcal{B}_{34}^{[4]}$
$\mathcal{B}_{1234}^{[1]}$ $\mathcal{B}_{1234}^{[2]}$ $\mathcal{B}_{1234}^{[3]}$ $\mathcal{B}_{1234}^{[4]}$

Orthogonalization

Any tensor $\mathcal{X}$ in $\mathcal{H}$-Tucker format can be orthogonalized in the sense that all factors in the dimension tree, except for the root node, contain orthonormal columns.

Example: $\text{vec}(\mathcal{X}) = (U_4 \otimes U_3 \otimes U_2 \otimes U_1)(B_{34} \otimes B_{12})B_{1234}$.

Step 1: QR decompositions $U_t = Q_t R_t \rightsquigarrow$

$$\text{vec}(\mathcal{X}) = (Q_4 \otimes Q_3 \otimes Q_2 \otimes Q_1)(\widetilde{B}_{34} \otimes \widetilde{B}_{12})B_{1234}$$

with $\widetilde{B}_{34} := (R_4 \otimes R_3)B_{34}$, $\widetilde{B}_{12} := (R_2 \otimes R_1)B_{12}$.

Step 2: QR decompositions $\widetilde{B}_{34} = Q_{34}R_{34}$, $\widetilde{B}_{12} = Q_{12}R_{12} \rightsquigarrow$

$$\text{vec}(\mathcal{X}) = (Q_4 \otimes Q_3 \otimes Q_2 \otimes Q_1)(Q_{34} \otimes Q_{12})\widetilde{B}_{1234}$$

with $\widetilde{B}_{1234} := (R_{34} \otimes R_{12})B_{1234}$.

Compt. requirements for general d: $\mathcal{O}(dnr^2) + \mathcal{O}(dr^4)$.

Command in `htucker`: `x = orthog(x)`

Norms and inner products

Inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{n_1 \times \cdots n_d}$:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \langle \mathrm{vec}(\mathcal{X}), \mathrm{vec}(\mathcal{Y}) \rangle = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \overline{x_{i_1,\dots,i_d}} y_{i_1,\dots,i_d}.$$

Can be performed efficiently in $\mathcal{H}$-Tucker, provided that $\mathcal{X}, \mathcal{Y}$ have compatible dimension trees.

Example: Two tensors of order 4 $\rightsquigarrow$

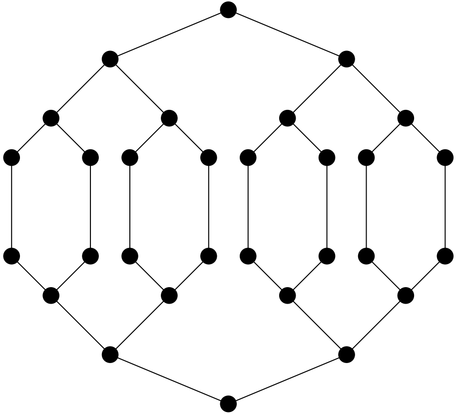
$$\begin{aligned} \langle \mathcal{X}, \mathcal{Y} \rangle \quad = \quad & (B^x_{1234})^T (B^x_{34} \otimes B^x_{12})^T (U^x_4 \otimes U^x_3 \otimes U^x_2 \otimes U^x_1)^T \cdots \\ & \cdots (U^y_4 \otimes U^y_3 \otimes U^y_2 \otimes U^y_1)(B^y_{34} \otimes B^y_{12}) B^y_{1234} \end{aligned}$$

Norm: *After* $\mathcal{X}$ has been orthogonalized:

$$\|\mathcal{X}\| = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle} = \|B^x_{12\cdots d}\|_F.$$

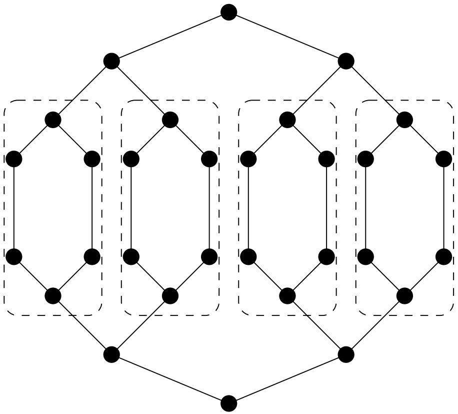
Possibly most accurate way to compute norm. Used in `norm(x)`.

Computation of inner products

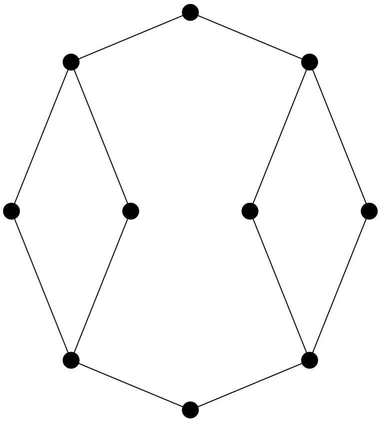


$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \overline{x_{i_1,\ldots,i_d}} y_{i_1,\ldots,i_d}.$$

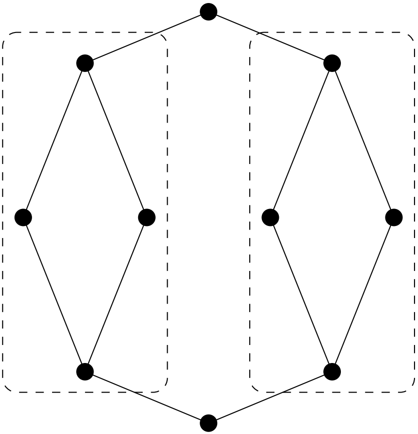
Computation of inner products



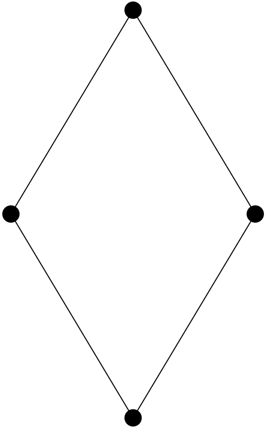
Computation of inner products



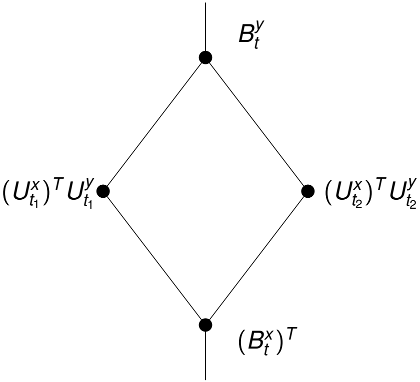
Computation of inner products



Computation of inner products



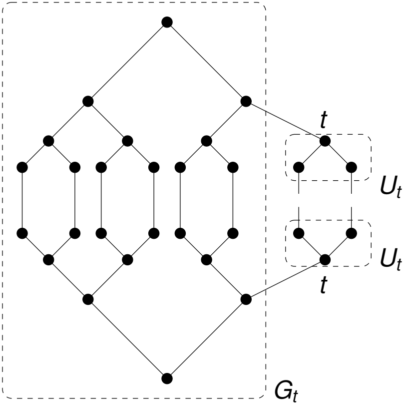
Computation of inner products – contraction step



$$(U_t^x)^T U_t^y = (B_t^x)^T \big((U_{t_2}^x)^T U_{t_2}^y \otimes (U_{t_1}^x)^T U_{t_1}^y\big) B_t^y.$$

- `htucker` command: `innerprod(x,y)`
- Overall cost: $\mathcal{O}(dnr^2) + \mathcal{O}(dr^4)$.

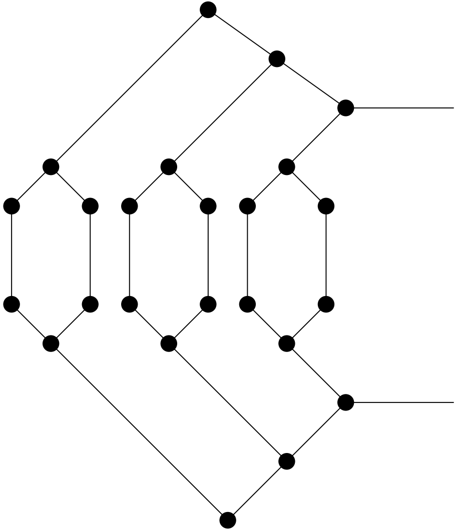
Reduced Gramians in $\mathcal{H}$-Tucker



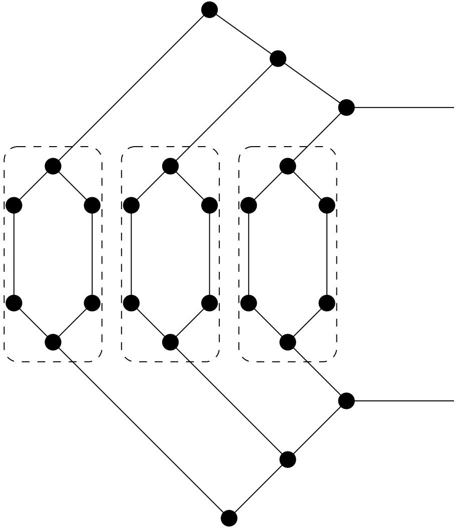
$$X^{(t)} = U_t V_t^T \quad \Rightarrow \quad X^{(t)} (X^{(t)})^T = U_t \underbrace{V_t^T V_t}_{=:G_t} U_t^T$$

If U_t orthonormal $\rightsquigarrow$ $\mathrm{svd}\big(X^{(t)}\big) = \sqrt{\mathrm{eig}(G_t)}$ (used in `plot_sv`).

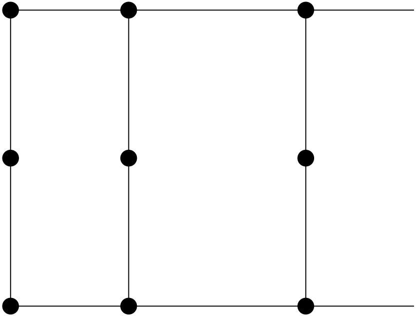
Reduced Gramians in $\mathcal{H}$-Tucker



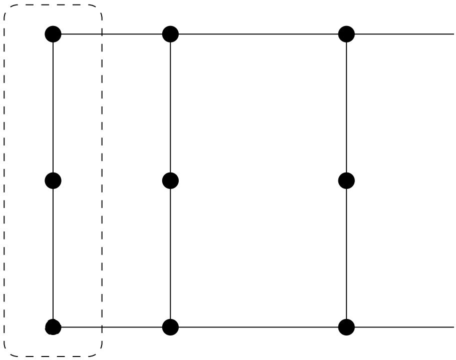
Reduced Gramians in $\mathcal{H}$-Tucker



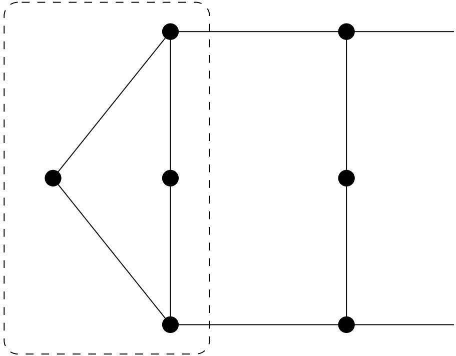
Reduced Gramians in $\mathcal{H}$-Tucker



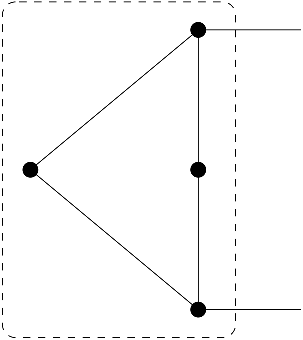
Reduced Gramians in $\mathcal{H}$-Tucker



Reduced Gramians in $\mathcal{H}$-Tucker



Reduced Gramians in $\mathcal{H}$-Tucker



Implemented in `htucker` command `gramians(x)`.

Advanced operations

- Truncation
- Combined addition + truncation
- Elementwise multiplication
- Elementwise reciprocal

Truncation of explicit tensor

Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be explicitly given.

- For each tree node t, let W_t contain r_t dominant left singular vectors of $X^{(t)}$ and define projection

$$\pi_t \mathcal{X} = W_t W_t^T \circ_t \mathcal{X} \qquad \Leftrightarrow \qquad \pi_t X^{(t)} = W_t W_t^T X^{(t)}.$$

- Truncated tensor:

$$\tilde{\mathcal{X}} := \Big(\prod_{t \in \mathcal{T}_L} \pi_t \Big) \cdots \Big(\prod_{t \in \mathcal{T}_1} \pi_t \Big) \mathcal{X},$$

  where $\mathcal{T}_\ell$ contains all nodes on level ℓ.
- [Grasedyck'2010]: $\|\mathcal{X} - \tilde{\mathcal{X}}\| \leq \sqrt{2d-3}\, \|\mathcal{X} - \mathcal{X}_{\text{best}}\|$.
  Proof similar as for HOSVD.

Truncation of explicit tensor

Example:

$$
\begin{aligned}
\mathrm{vec}\tilde{\mathcal{X}} &= (W_4 W_4^T \otimes W_3 W_3^T \otimes W_2 W_2^T \otimes W_1 W_1^T)(W_{34} W_{34}^T \otimes W_{12} W_{12}^T)\mathrm{vec}\mathcal{X} \\
&= (W_4 \otimes W_3 \otimes W_2 \otimes W_1) \cdots \\
&\quad (\underbrace{[W_4^T \otimes W_3^T] W_{34}}_{=:B_{34}} \otimes \underbrace{[W_2^T \otimes W_1^T] W_{12})}_{=:B_{12}} \underbrace{([W_{34}^T \otimes W_{12}^T]\mathrm{vec}\mathcal{X})}_{=:B_{1234}}.
\end{aligned}
$$

`opts.max_rank = 10` maximal rank at truncation.
`opts.rel_eps = 1e-6` maximal relative truncation error.
`opts.abs_eps = 1e-6` maximal absolute truncation error.
Condition `max_rank` takes precedence over `rel_eps` and `abs_eps`.
`xt = htensor.truncate_rtl(x, opts)` returns truncated tensor $\tilde{\mathcal{X}}$ of a multidimensional array.

Remark: There is also a significantly faster `htensor.truncate_ltr` (proceeds successively from leafs to roots), for which the same error bound holds [Tobler'10].

Truncation of $\mathcal{H}$-Tucker tensor

Let $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ be in $\mathcal{H}$-Tucker format and *orthogonalized*.

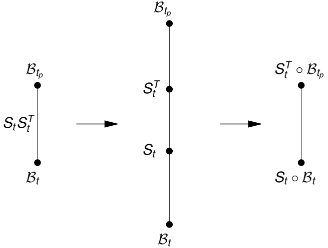
- Compute left singular vectors of $X^{(t)} = U_t V_t^T$ from eigenvectors of

$$X^{(t)} \left(X^{(t)}\right)^T = U_t \underbrace{V_t^T V_t}_{=G_t} U_t^T,$$

  with reduced Gramian G_t.

  If S_t contains r_t dominant eigenvectors of $G_t \rightsquigarrow W_t = U_t S_t$.

- Traverse tree from root to leafs. In each step:



- In `htucker`: `truncate(x,opts)`. Complexity $\mathcal{O}(dnr^2 + dr^4)$.

Combined addition + truncation

Sum of more than two tensors:

$$\mathcal{Y} = \mathcal{X}_1 + \mathcal{X}_2 + \cdots + \mathcal{X}_s.$$

Two possibilities to incorporate truncation operator $\mathcal{T}$:

1. $\mathcal{Y} \approx \mathcal{T}(\mathcal{X}_1 + \mathcal{X}_2 + \mathcal{X}_3 + \cdots + \mathcal{X}_s)$
2. $\mathcal{Y} \approx \mathcal{T}(\cdots(\mathcal{T}(\mathcal{T}(\mathcal{X}_1 + \mathcal{X}_2) + \mathcal{X}_3) + \cdots + \mathcal{X}_s)$

Option 2 is usually significantly cheaper but may suffer from severe cancellation.

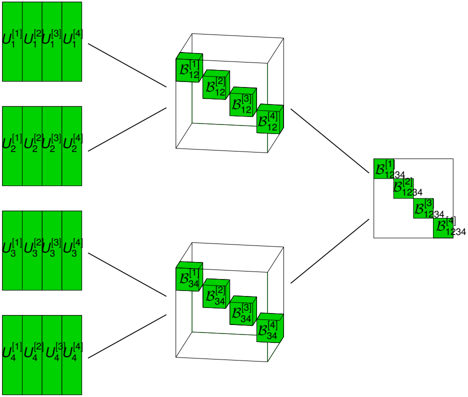
Artificial example: $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \in \mathbb{R}^{101\times101\times101}$ truncated tensor grid discretizations for summands of

$$f(x_1, x_2, x_3) = \tan(x_1 + x_2 + x_3) + (x_1 + x_2 + x_3)^{-1} - \tan(x_1 + x_2 + x_3).$$

Error(Option 1) $\approx 10^{-7}$. Error(Option 2) ≈ 1.3.

What is wrong with Option 1?

Combined addition + truncation

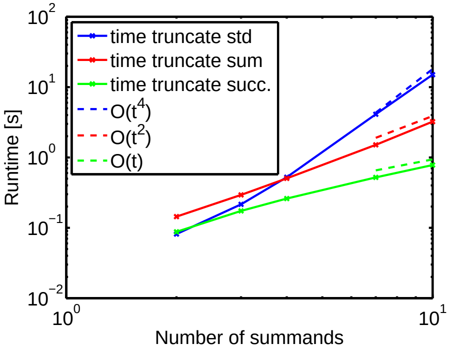


- Orthogonalization (needed before truncation) destroys block diagonal structure.
- Complexity $\mathcal{O}(dns^2r^2 + ds^4r^4)$ for s summands.

Combined addition + truncation

Idea: New variant delays orthogonalization to keep block diagonal structure in transfer tensors as long as possible.

Reduces $\mathcal{O}(dns^2r^2 + ds^4r^4)$ to $\mathcal{O}(dns^2r^2 + ds^2r^4 + ds^3r^3)$



- htucker command: `add_truncate(x1 x2 x3 x4, opts)`.

Elementwise multiplication

Elementwise multiplication (also called Hadamard or Schur product) of two low-rank matrices $A = U_1 \Sigma_A U_2^T$, $B = V_1 \Sigma_B V_2^T$:

$$A \star B = (U_1 \mathbin{\tilde{\odot}} V_1)(\Sigma_A \otimes \Sigma_B)(U_2 \mathbin{\tilde{\odot}} V_2)^T,$$

with the row-wise Khatri-Rao product

$$C \mathbin{\tilde{\odot}} D = \begin{bmatrix} c_1^T \\ \vdots \\ c_n^T \end{bmatrix} \tilde{\odot} \begin{bmatrix} d_1^T \\ \vdots \\ d_n^T \end{bmatrix} = \begin{bmatrix} c_1^T \otimes d_1^T \\ \vdots \\ c_n^T \otimes d_n^T \end{bmatrix}$$

- Orthogonality destroyed.
- Rank increases *significantly*.

But: singular value decay of $\Sigma_A \otimes \Sigma_B$ may become significantly stronger $\rightsquigarrow$ additional opportunities for truncation.

Elementwise multiplication

Elementwise multiplication of two tensors $\mathcal{X}, \mathcal{Y}$ in $\mathcal{H}$-Tucker format:

- Row-wise Khatri-Rao product of leaf matrices.
- "Kronecker product" of non-leaf tensors.
- Optional: Products are only formed after suitable truncation to avoid excessive memory requirements.

Commands in `htucker`:

`x.*y` (without truncation)

`x.^2` (without truncation)

`elem_mult( x, y, opt )` (with truncation)

Elementwise reciprocal

Goal: Compute reciprocal of each entry in tensor $\mathcal{X}$.

Basic idea: Newton-Schultz iteration

$$y_0 = 1, \quad y_{i+1} = y_i + y_i(1 - x\, y_i), \tag{2}$$

converges to $1/x$ for $0 < x < 2$.

Apply (2) simultaneously to all entries.

Code snippet of `elem_reciprocal( x, opt )` in `htucker`:

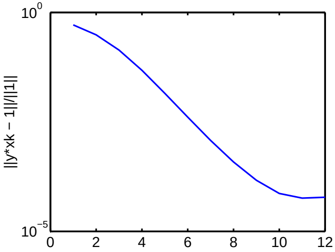
```
all_ones = htenones(size(x));
y = all_ones;
for it=1:maxit
  xy = elem_mult( x, y );
  xy =  truncate( all_ones - xy );
  xy = elem_mult( xy, y );
  y  =  truncate( y + xy );
end
```

See also [Oseledets et al. 2009].

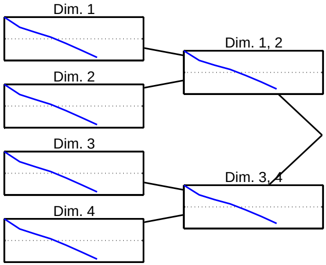
Elementwise reciprocal

Example: $(x_1 + x_2 + x_3 + x_4)^{-1}$ with $x_i \in [10^{-3}, 1]$.

```
c = laplace_core(4);
U = [ones(100, 1), linspace(1e-3, 1, 100)'];
x = ttm(c, {U, U, U, U});
inv_x = elem_reciprocal(x, opts);
```



Convergence of $\|\mathcal{X} \star \mathcal{Y}_k - 1\|$.



Singular value tree upon convergence.

Summary

- HTD offers good compromise between CP and Tucker.
- Algorithms often quite technical but conceptually simple.
- Computational complexity $\sim d$ but often $\sim r^4$:

  Curse of dimensionality $\Rightarrow$ curse of rank ?

- Important to keep in mind:

  Unless d is tiny, tensor $\mathcal{X}$ can/should never be formed explicitly.

  All operations need to be performed implicitly in HTD.

  Can pose severe problems even for seemingly simple operations: $\min(\mathcal{X})$, $\max(\mathcal{X})$, $\text{abs}(\mathcal{X})$, $1./\mathcal{X}$, $\ldots$

Algorithms based on low-rank tensors

- Inexact LOBPCG
- ALS / MALS

Strategies for solving tensor equations

- In many practical situations, tensor $\mathcal{X}$ is given *implicitly* as solution to linear system $\mathcal{A}(\mathcal{X}) = \mathcal{B}$, eigenvalue problem $\mathcal{A}(\mathcal{X}) = \lambda\mathcal{X}$, nonlinear system, ODE, ...

Two main strategies to use low-rank tensor techniques:

1. Combine existing iterative solver (e.g., CG, LOBPCG, GMRES) with repeated low-rank truncation of iterates ($\rightsquigarrow$ inexact CG).
   - Straightforward to derive and implement (based, e.g., on `htucker`).
   - Hard to analyze impact of nonnegligible truncations on accuracy and convergence.
   - Intermediate rank growth may result in excessive computing times and/or harm accuracy+convergence.
2. Formulate optimization problem, constrain to low-rank tensors, iteratively optimize wrt individual factors of low-rank format.
   - Works well in practice.
   - Convergence theory not well understood.
   - *Not* straightforward to implement.

Example: PDE-eigenvalue problem

Goal: Compute smallest eigenvalue for

$$\begin{aligned} \Delta u(\xi) + V(\xi)u(\xi) &= \lambda\, u(\xi) \qquad \text{in } \Omega = [0,1]^d, \\ u(\xi) &= 0 \qquad \text{on } \partial\Omega. \end{aligned}$$

Assumption: Potential represented as

$$V(\xi) = \sum_{j=1}^{s} V_j^{(1)}(\xi_1)\, V_j^{(2)}(\xi_2) \cdots V_j^{(d)}(\xi_d).$$

$\rightsquigarrow$ finite difference discretization

$$\mathcal{A}\mathbf{u} = (\mathcal{A}_L + \mathcal{A}_V)\mathbf{u} = \lambda \mathbf{u},$$

with

$$\begin{aligned} \mathcal{A}_L &= \sum_{j=1}^{d} \underbrace{I \otimes \cdots \otimes I}_{d-j \text{ times}} \otimes A_L \otimes \underbrace{I \otimes \cdots \otimes I}_{j-1 \text{ times}}, \\ \mathcal{A}_V &= \sum_{j=1}^{s} A_{V,j}^{(d)} \otimes \cdots \otimes A_{V,j}^{(2)} \otimes A_{V,j}^{(1)}. \end{aligned}$$

LOBPCG method

LOBPCG with block size 1 [Knyazev'01] for computing smallest eigenvalue of

$$Ax = \lambda x, \qquad A \text{ symmetric.}$$

$\lambda_0 = \langle x_0, x_0 \rangle_A$, $p_0 = 0$
for $k = 0, 1, \ldots$ (until converged) **do**
  $r_k = B^{-1}(Ax_k - \lambda_k x)$
  $U = \left[x_k,\, r_k,\, p_k\right]$
  $\tilde{A} = U^T A U,\ \tilde{M} = U^T U$
  Find eigenpair (λ_{k+1}, y), with $\|y\|_2 = 1$, for smallest eigenvalue of matrix pencil $\tilde{A} - \lambda \tilde{M}$.
  $p_{k+1} = y_2 \cdot r_k + y_3 \cdot p_k$
  $x_{k+1} = y_1 \cdot x_k + p_{k+1}$
  $x_{k+1} \leftarrow x_{k+1} / \|x_{k+1}\|_2$
end for
Return $(\lambda_{\min}, x) = (\lambda_{k+1}, x_{k+1})$.

Tensor low-rank LOBPCG

Truncated LOBPCG with block size 1 for computing smallest eigenvalue of

$$\mathcal{A}(\mathcal{X}) = \lambda \mathcal{X}, \qquad \mathcal{A} \text{ symmetric, } \mathcal{X} \text{ tensor.}$$

$\lambda_0 = \langle \mathcal{X}_0, \mathcal{X}_0 \rangle_{\mathcal{A}}$, $\mathcal{P}_0 = 0 \cdot \mathcal{X}$

for $k = 0, 1, \ldots$ (until converged) **do**

$\mathcal{R}_k = \mathcal{B}^{-1}(\mathcal{A}(\mathcal{X}_k) - \lambda_k \mathcal{X}_k)$, $\qquad \mathcal{R}_k \leftarrow \mathcal{T}(\mathcal{R}_k)$

$\mathcal{U}_1 = \mathcal{X}_k$, $\mathcal{U}_2 = \mathcal{R}_k$, $\mathcal{U}_3 = \mathcal{P}_k$

$\tilde{A}_{ij} = \langle \mathcal{U}_i, \mathcal{U}_j \rangle_{\mathcal{A}}$, $\tilde{M}_{ij} = \langle \mathcal{U}_i, \mathcal{U}_j \rangle$

Find eigenpair (λ_{k+1}, y), with $\|y\|_2 = 1$, for smallest eigenvalue of matrix pencil $\tilde{A} - \lambda \tilde{M}$.

$\mathcal{P}_{k+1} = y_2 \cdot \mathcal{R}_k + y_3 \cdot \mathcal{P}_k$ $\qquad \mathcal{P}_{k+1} \leftarrow \mathcal{T}(\mathcal{P}_{k+1})$

$\mathcal{X}_{k+1} = y_1 \cdot \mathcal{X}_k + \mathcal{P}_{k+1}$ $\qquad \mathcal{X}_{k+1} \leftarrow \mathcal{T}(\mathcal{X}_{k+1})$

$\mathcal{X}_{k+1} \leftarrow \mathcal{X}_{k+1} / \sqrt{\langle \mathcal{X}_{k+1}, \mathcal{X}_{k+1} \rangle}$

end for

Return $(\lambda_{\min}, \mathcal{X}) = (\lambda_{k+1}, \mathcal{X}_{k+1})$.

$\mathcal{T} =$ truncation to hierarchical low rank

Implementation details

Orthogonalization
In standard LOBPCG, orthogonalization of U is recommended [Knyazev 2010]. This is not practical with low-rank tensors, as ranks would grow and truncation would destroy orthogonality.

Truncation
$\mathcal{X}_k, \mathcal{R}_k, \mathcal{P}_k$ are truncated in every step. Moreover, application of $\mathcal{A}(\cdot)$ and preconditioner $\mathcal{B}^{-1}(\cdot)$ may involve truncation during the application of these operators.

Inner product
Reduced matrix $\tilde{A}$ is very sensitive to truncation in $\mathcal{A}(\cdot)$. The computation of $\tilde{A}_{i,j} = \langle \mathcal{U}_i, \mathcal{U}_j \rangle_{\mathcal{A}}$ must be exact.

Numerical Experiments - Sine potential

PDE-eigenvalue problem with $\Omega = [0, \pi]^d$ and sine potential

$$V(\xi) = q \cdot \prod_{i=1}^{d} \sin(\xi_i)$$

for some constant $q > 0$. We choose $d = 10$, $n = 128$.

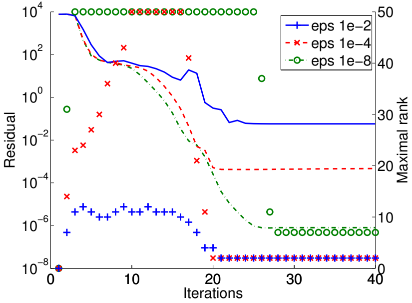
Preconditioner: [Grasedyck 2004]

$$\begin{aligned}\mathcal{A}_L^{-1} &= \int_0^\infty \exp(-t\mathcal{A}_L)\mathrm{d}t \\ &\approx \sum_{j=-M}^{M} \omega_j \exp(-\alpha_j A_L^{(d)}) \otimes \cdots \otimes \exp(-\alpha_j A_L^{(1)}) =: \mathcal{B}^{-1},\end{aligned}$$

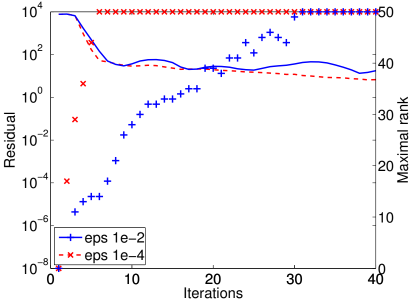
for a certain, optimized and tabulated choice of coefficients $\alpha_j, \omega_j > 0$.
We choose $M = 10$.

Numerical Experiments - Sine potential

$q = 1$



$q = 1000$



ALS

Originally from computational quantum physics [Schollwöck 2011], recently investigated by [Huckle et al. 2010; Oseledets, Khoromskij 2010; Holtz et al. 2010; Dolgov, Oseledets 2011]

Goal:

$$\min\left\{\frac{\langle\mathcal{X},\mathcal{A}(\mathcal{X})\rangle}{\langle\mathcal{X},\mathcal{X}\rangle}\,:\,\mathcal{X}\in\mathcal{H}\text{-Tucker}\big((r_t)_{t\in\mathcal{T}}\big),\ \mathcal{X}\neq 0\right\}$$

Method: Choose one node t, fix all other nodes, set new tensor at node t to minimize Rayleigh quotient $\frac{\langle\mathcal{X},\mathcal{A}(\mathcal{X})\rangle}{\langle\mathcal{X},\mathcal{X}\rangle}$. This is done for all nodes (a sweep), and sweeps are continued until convergence.

Sketch:

$$\begin{aligned} X^{(t)} &= U_t V_t^T = \big(U_{t_r}\otimes U_{t_l}\big)B_t V_t^T,\\ \operatorname{vec}(\mathcal{X}) &= \big(V_t\otimes U_{t_r}\otimes U_{t_l}\big)\operatorname{vec}(B_t) = \mathcal{U}_t\,\operatorname{vec}(B_t). \end{aligned}$$

$$\Rightarrow \min\left\{\frac{y^T(\mathcal{U}_t^T\mathcal{A}\mathcal{U}_t)y}{y^T(\mathcal{U}_t^T\mathcal{U}_t)y}\,:\,y\in\mathbb{R}^{r_{t_l}r_{t_r}r_t},\ y\neq 0\right\}.$$

Computation of reduced matrices

Consider $\mathcal{A} = A_d \otimes \cdots \otimes A_1$ (Other operators can be treated similarly)

Compute

$$\tilde{\mathcal{A}}_t := \mathcal{U}_t^T \mathcal{A} \mathcal{U}_t = \left(V_t \otimes U_{t_r} \otimes U_{t_l}\right)^T \mathcal{A}\left(V_t \otimes U_{t_r} \otimes U_{t_l}\right) = \hat{A}_t \otimes \tilde{A}_{t_r} \otimes \tilde{A}_{t_l},$$

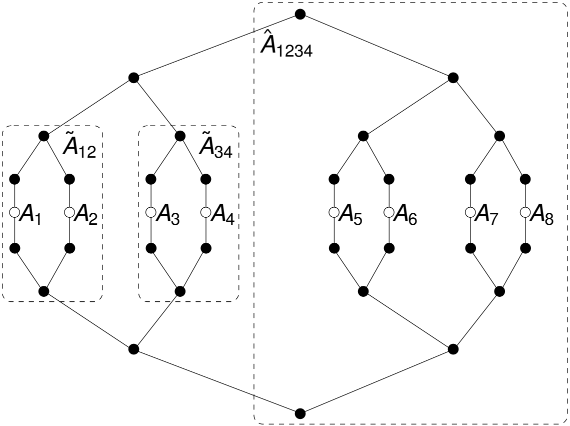
where

$$\tilde{A}_{t_l} = U_{t_l}^T \Big(\bigotimes_{i \in t_l} A_i \Big) U_{t_l}, \quad \tilde{A}_{t_r} = U_{t_r}^T \Big(\bigotimes_{i \in t_r} A_i \Big) U_{t_r}, \quad \hat{A}_t = V_t^T \Big(\bigotimes_{i \notin t} A_i \Big) V_t.$$

Additionally

$$\tilde{\mathcal{M}}_t := \mathcal{U}_t^T \mathcal{U}_t = V_t^T V_t \otimes U_{t_r}^T U_{t_r} \otimes U_{t_l}^T U_{t_l} = M_t \otimes M_{t_r} \otimes M_{t_l},$$

Computation of reduced matrices

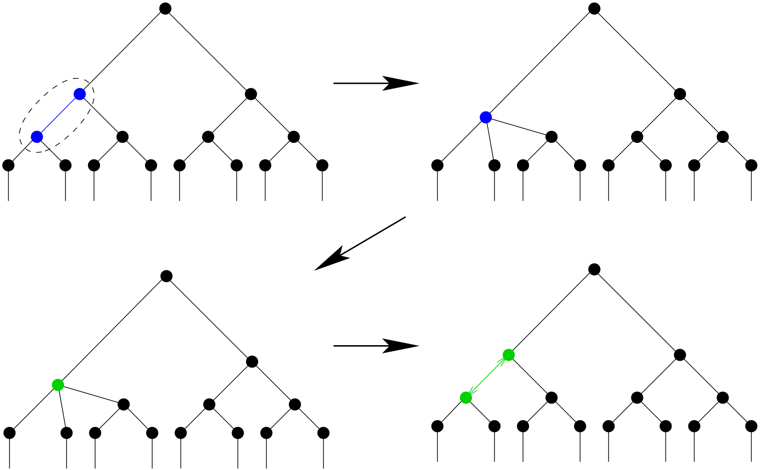


MALS

Method:

- Select edge of tensor network.
- Combine tensors at the adjacent nodes to form a higher-order tensor.
- Set this tensor to minimize the Rayleigh quotient.
- Use low-rank approximation to split new combined tensor into two tensors at adjacent nodes of selected edge.

MALS - Illustration



Numerical Experiments – Sine potential

PDE-eigenvalue problem with $\Omega = [0, \pi]^d$ and sine potential

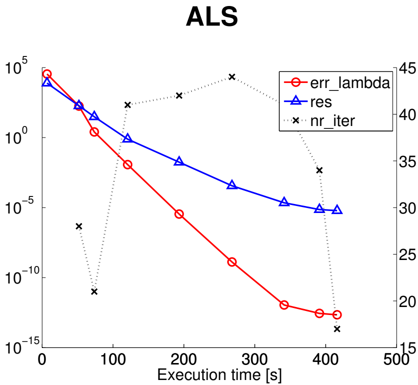
$$V(\xi) = q \cdot \prod_{i=1}^{d} \sin(\xi_i)$$

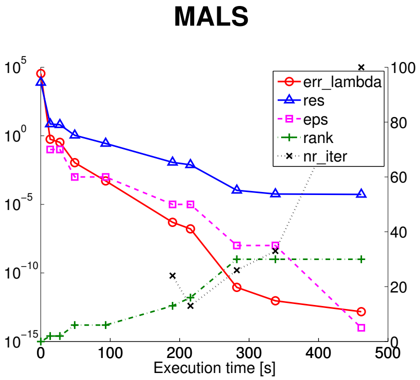
for some constant $q > 0$. Choose $d = 10$, $n = 128$, $q = 1000$.
Preconditioner: [Grasedyck 2004]

$$\begin{aligned} \mathcal{A}_L^{-1} &= \int_0^\infty \exp(-t\mathcal{A}_L)\mathrm{d}t \\ &\approx \sum_{j=-M}^{M} \omega_j \exp(-\alpha_j A_L^{(d)}) \otimes \cdots \otimes \exp(-\alpha_j A_L^{(1)}) =: \mathcal{B}^{-1}, \end{aligned}$$

for a certain, optimized choice of coefficients $\alpha_j, \omega_j > 0$. We choose $M = 10$.

Numerical Experiments – Sine potential





Hierarchical ranks 40.

Maximal hierarchical rank 30.

Conclusions and Outlook

Conclusions and Outlook

- Scientific computing with low-rank tensors rapidly evolving field and highly technical.
- Precise scope of applications far from clear; many applications remain to be explored. More analysis and comparison to alternative techniques (sparse grids, adaptive tensor discretization, Monte Carlo, ...) needed.

Some current trends:

- Tensorization of vectors $+$ low rank (discrete Chebfun?) by Hackbusch, Khoromskij, Oseledets, Tyrtishnikov, ...
- Computational differential geometry on low-rank tensor manifolds by Koch, Lubich, Schneider, Uschmajew, Vandereycken, ...
- Robust low rank (Candes et al.) for tensors $\rightsquigarrow$ suitable way of dealing with singularities?
- ...

Acknowledgments: Presentation heavily benefited from joint work with Christine Tobler (ETH Zurich).