# Low Rank Approximation
# Lecture 4

Daniel Kressner
Chair for Numerical Algorithms and HPC
Institute of Mathematics, EPFL
daniel.kressner@epfl.ch

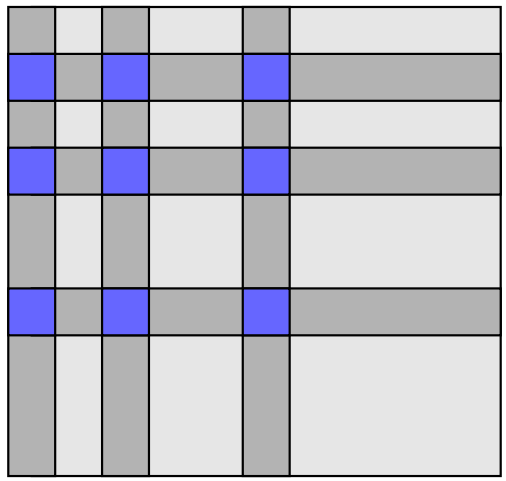

# Plan

- A peek into randomized column/row sampling algorithms.
- The Kronecker product
- First steps with tensors
- Applications of tensors

# Randomized Sampling

# Randomized column/row sampling

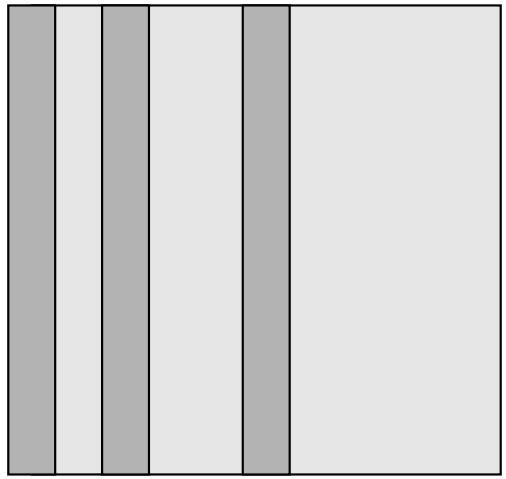
Aim: Obtain rank-$r$ approximation from randomly selected rows and columns of $A$.



Popular sampling strategies:

- Uniform sampling.
- Sampling based on row/column norms.
- Sampling based on more complicated quantities.

# Column sampling



Basic algorithm aiming at rank-$r$ approximation:

1. Sample (and possibly rescale) $k > r$ columns of $A$
   $\rightsquigarrow m \times k$ matrix $C$.
2. Compute SVD $C = U\Sigma V^T$ and set $Q = U_r \in \mathbb{R}^{m\times r}$.
3. Return low-rank approximation $QQ^T A$.

- Can be combined with streaming algorithm [Liberty'2007] to limit memory/cost of working with $C$.
- Quality of approximation crucially depends on sampling strategy.

# Column sampling

## Lemma

*For any matrix $C \in \mathbb{R}^{m\times r}$, let $Q$ be the matrix computed above. Then*

$$\|A - QQ^T A\|_2^2 \leq \sigma_{r+1}(A)^2 + 2\|AA^T - CC^T\|_2.$$

*Proof.* We have

$$\begin{aligned} & (A - QQ^T A)(A - QQ^T A)^T \\ = \; & (I - QQ^T)CC^T(I - QQ^T) + (I - QQ^T)(AA^T - CC^T)(I - QQ^T) \end{aligned}$$

Hence,

$$\begin{aligned} \|A - QQ^T A\|_2^2 &= \lambda_{\max}\big((A - QQ^T A)(A - QQ^T A)^T\big) \\ &\leq \lambda_{\max}\big((I - QQ^T)CC^T(I - QQ^T)\big) + \|AA^T - CC^T\|_2 \\ &= \sigma_{r+1}(C)^2 + \|AA^T - CC^T\|_2. \end{aligned}$$

The proof is completed by applying Weyl's inequality:

$$\sigma_{r+1}(C)^2 = \lambda_{r+1}(CC^T) \leq \lambda_{r+1}(AA^T) + \|AA^T - CC^T\|_2.$$

# Random column sampling

Using the lemma, the goal now becomes to approximate the matrix product $AA^T$ using column samples of $A$.

*Notation:*

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}, \quad C = \begin{bmatrix} c_1 & \cdots & c_k \end{bmatrix}$$

*General sampling method:*

**Input:** $A \in \mathbb{R}^{m \times n}$, probabilities $p_1, \ldots, p_n$, integer $k$.
**Output:** $C \in \mathbb{R}^{m \times k}$ containing selected columns of $A$.

1: **for** $t = 1, \ldots, k$ **do**
2: Pick $j_t \in \{1, \ldots, n\}$ with $\mathbb{P}[j_t = \ell] = p_\ell$, $\ell = 1, \ldots, n$, independently and with replacement.
3: Set $c_t = a_{j_t} / \sqrt{k p_{j_t}}$.
4: **end for**

EFY. Implement the above procedure in MATLAB using `rand`.

# Random column sampling

## Lemma

*For the matrix $C$ returned by algorithm, it holds that*

$$\mathbb{E}[CC^T] = AA^T, \quad \text{Var}[(CC^T)_{ij}] = \frac{1}{k}\sum_{\ell=1}^{n} \frac{a_{i\ell}^2 a_{j\ell}^2}{p_\ell} - \frac{1}{k}(AA^T)_{ij}^2.$$

*Proof.* For fixed $i, j$, consider $X_t = \left(c_t c_t^T\right)_{ij} = \frac{1}{kp_{j_t}} a_{i,j_t} a_{j,j_t}$, for which

$$\mathbb{E}[X_t] = \sum_{\ell=1}^{n} p_\ell \frac{1}{kp_\ell} a_{i,\ell} a_{j,\ell} = \frac{1}{k}(AA^T)_{ij}.$$

Analogously,

$$\text{Var}(X_t) = \mathbb{E}[(X_t - \mathbb{E}[X_t])^2] = \mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2 = \frac{1}{k^2}\sum_{\ell=1}^{n} \frac{a_{i\ell}^2 a_{j\ell}^2}{p_\ell} - \frac{1}{k^2}(AA^T)_{ij}^2.$$

Because of independence, it follows $\mathbb{E}[\sum_t X_t] = k \cdot \mathbb{E}[X_t] = (AA^T)_{ij}$, and analogously for variance.

# Random column sampling

As a consequence of the lemma,

$$\begin{aligned}
\mathbb{E}[\|AA^T - CC^T\|_F^2] &= \sum_{ij} \mathbb{E}[(AA^T - CC^T)_{ij}^2] \\
&= \sum_{ij} \mathrm{Var}[(CC^T)_{ij}] \\
&= \frac{1}{k} \sum_{ij} \left( \sum_{\ell=1}^{n} \frac{a_{i\ell}^2 a_{j\ell}^2}{p_\ell} - \frac{1}{k} (AA^T)_{ij}^2 \right) \\
&= \frac{1}{k} \left[ \sum_{\ell=1}^{n} \frac{1}{p_\ell} \|a_\ell\|_2^4 - \|AA^T\|_F^2 \right].
\end{aligned}$$

## Lemma

*The choice* $p_\ell = \|a_\ell\|_2^2 / \|A\|_F^2$ *minimizes* $\mathbb{E}[\|AA^T - CC^T\|_F^2]$ *and yields*

$$\mathbb{E}[\|AA^T - CC^T\|_F^2] = \frac{1}{k} \left[ \|A\|_F^4 - \|AA^T\|_F^2 \right]$$

*Proof.* Established by showing that this choice of $p_\ell$ satisfies first-order conditions of constrained optimization problem.

# Random column sampling

*Norm based sampling:*

**Input:** $A \in \mathbb{R}^{m \times n}$, integer $k$.

**Output:** $C \in \mathbb{R}^{m \times k}$ containing selected columns of $A$.

1: Set $p_\ell = \|a_\ell\|_2^2 / \|A\|_F^2$ for $\ell = 1, \ldots, n$.
2: **for** $t = 1, \ldots, k$ **do**
3: Pick $j_t \in \{1, \ldots, n\}$ with $\mathbb{P}[j_t = \ell] = p_\ell$, $\ell = 1, \ldots, n$, independently and with replacement.
4: Set $c_t = a_{j_t} / \sqrt{k p_{j_t}}$.
5: **end for**
5: Compute SVD $C = U \Sigma V^T$ and set $Q = U_r \in \mathbb{R}^{m \times r}$.
5: Return low-rank approximation $QQ^T A$.

# Random column sampling

## Lemma

*For the matrix C returned by algorithm, it holds with probability* $1-\delta$ *that*

$$\|AA^T - CC^T\|_F \leq \frac{\eta}{\sqrt{k}}\|A\|_F,$$

*where* $\eta = 1 + \sqrt{8 \cdot \log(1/\delta)}$.

*Proof.* Aim at applying Azuma-Hoeffding inequality. Define

$$F(i_1, i_2, \ldots, i_k) = \|AA^T - CC^T\|_F,$$

with $C = \begin{bmatrix} a_{i_1} & \cdots & a_{i_k} \end{bmatrix}$. Quantify the effect of varying an index (w.l.o.g. the first one) on $F$:

$$\begin{aligned} & |F(i_1, i_2, \ldots, i_k) - F(i_1', i_2, \ldots, i_k)| \\ = & \left| \|AA^T - CC^T\|_F - \|AA^T - C'C'^T\|_F \right| \\ \leq & \|CC^T - C'C'^T\|_F \leq \frac{1}{kp_{i_1}}\|a_{i_1}\|_2^2 + \frac{1}{kp_{i_1'}}\|a_{i_1'}\|_2^2 \\ \leq & \frac{2}{k}\|A\|_F^2 := \Delta. \end{aligned}$$

# Random column sampling

This implies that Doob martingales $g_\ell = \mathbb{E}[f(i_1, \ldots, i_k)|i_1, \ldots, i_\ell]$ for $1 \leq \ell \leq k$ satisfy

$$|g_{\ell+1} - g_\ell| \leq \Delta.$$

Note that $g_k = \mathbf{E}[\|AA^T - CC^T\|_F]$. By lemma and Jensen's inequality we know that $g_k \leq \|A\|_F^2/\sqrt{k}$. Applying Azuma-Hoeffding inequality to $g_k$ yields

$$P\left[\|AA^T - CC^T\|_F \geq \|A\|_F^2/\sqrt{k} + \gamma\right] \leq \exp(-\gamma^2/2k\Delta^2) =: \delta.$$

Setting $\gamma = \sqrt{8 \cdot \log(1/\delta)}$ completes the proof.

# Random column sampling

## Theorem (Drineas/Kannan/Mahoney'2006)

*For the matrix Q returned by the algorithm above it holds that*

$$\mathbf{E}\left[\|A - QQ^T A\|_2^2\right] \leq \sigma_{r+1}^2(A) + \varepsilon\|A\|_F^2 \text{ for } k \geq 4/\varepsilon^2.$$

*With probability at least* $1 - \delta$,

$$\|A - QQ^T A\|_2^2 \leq \sigma_{r+1}^2(A) + \varepsilon\|A\|_F^2 \text{ for } k \geq 4(1 + \sqrt{8 \cdot \log(1/\delta)})^2/\varepsilon^2.$$

*Proof.* Follows from combining very first lemma with last two lemmas.

*Remarks:*

- Dependence of $k$ on $\varepsilon$ pretty bad. Unlikely to achieve something significantly better without assuming further properties of $A$ (e.g., incoherence of singular vectors) with sampling based on row norms only.
- Simple "counter example":

$$A = \left(\frac{1}{\sqrt{n}}e_1 \quad \frac{1}{\sqrt{n}}e_1 \quad \cdots \quad \frac{1}{\sqrt{n}}e_1 \quad \frac{1}{\sqrt{n}}e_2\right) \in \mathbb{R}^{n\times(n+1)}.$$

# Random column sampling

[Drineas/Mahoney/Muthukrishnan'2007]: Let $V_k$ contain $k$ dominant right singular vectors of $A$. Setting

$$p_\ell = \|V_k(\ell,:)\|_2^2/k, \qquad \ell = 1,\ldots,n$$

and sampling $\mathcal{O}(k^2(\log 1/\delta)/\varepsilon^2)$ columns[1] yields

$$\|A - QQ^TA\|_F \leq (1+\varepsilon)\|A - \mathcal{T}_k(A)\|_F$$

with probability $1-\delta$.

Relative error bound!

EFY. Implement both sampling strategies in Matlab. Try to find an example for which the strategy based on $\|V_k(\ell,:)\|_2$ performs much better. Hint: Make the norms $\|V_k(\ell,:)\|_2$ very different while keeping column norms balanced.

CUR decomposition can be obtained by applying ideas to rows and columns (yielding $R$ and $C$, respectively) and choosing $U$ appropriately.

[1] There are variants that improve this to $\mathcal{O}(k\log k\log(1/\delta)/\varepsilon^2)$.

# Further literature on random sampling

- Michael W. Mahoney. Randomized algorithms for matrices and data. Foundations and Trends in Machine Learning, 3(2):123–224, 2011.
- D. P. Woodruff. Sketching as a Tool for Numerical Linear Algebra. Foundations and Trends in Theoretical Computer Science, vol. 10, no. 1-2, pp. 1–157, 2014

# The Kronecker product

# Vectorization

The vectorization of an $m \times n$ matrix $A$ denoted by $\text{vec}(A)$, where

$$\text{vec} : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \cdot n}$$

stacks the columns of a matrix into a long column vector.

Example:

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

Remarks:

- In MATLAB: `A(:)`
- This way of vectorizing corresponds to how matrices are laid out in memory in MATLAB. In other programming languages (e.g., C arrays) matrices are laid out rowwise.

# 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. $\text{vec}(AX) = (I \otimes A)\,\text{vec}(X)$.
2. $\text{vec}(XA^T) = (A \otimes I)\,\text{vec}(X)$.
3. $(B \otimes A)(D \otimes C) = (BD \otimes AC)$
   for $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{k \times \ell}$, $C \in \mathbb{R}^{n \times q}$, $D \in \mathbb{R}^{\ell \times p}$.
4. $I_m \otimes I_n = I_{mn}$.
5. $(A_1 + A_2) \otimes B = A_1 \otimes B + A_2 \otimes B$, $A \otimes (B_1 \otimes B_2) = A \otimes B_1 + A \otimes B_2$

EFY. Write $(A \otimes B)^T$ as a Kronecker product of two matrices.

EFY. Show how the SVD of $B \times A$ can be obtained from the SVDs of $A$, $B$.

EFY. The Lyapunov matrix equation, which plays an important role in dynamical systems and control, takes the form $AX + XA^T = C$, where $A$ and $C$ are $n \times n$ known coefficient matrices and $X$ is the $n \times n$ solution matrix. Using Kronecker products, reformulate this matrix equation as a linear system.

# Kronecker product

Reference:

- Van Loan, C. F. The ubiquitous Kronecker product. J. Comput. Appl. Math. 123 (2000), no. 1-2, 85–100.

To get familiar with Kronecker products, try the following problem:

EFY. [Van Loan'2000] explains how low-rank matrix approximation can be used to solve the following approximation problem:

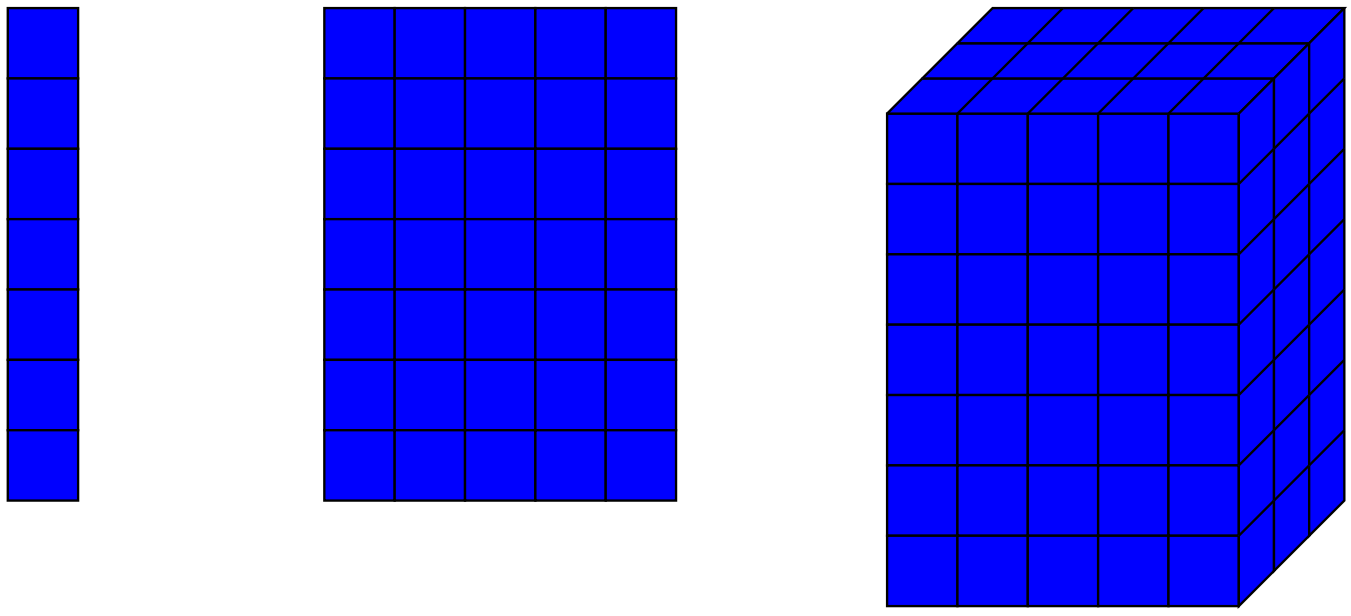
$$\min \left\| A - \sum_{j=1}^{r} C_j \otimes B_j \right\|_F$$

for a given $n^2 \times n^2$ matrix and unknown matrices $B_j, C_j \in \mathbb{R}^{n \times n}$. Implement this procedure in MATLAB using `svd`.
(Hint: The most elegant approach is to view $A$ as a fourth-order tensor and permute its dimensions; in MATLAB: `permute`, `reshape`.)

# First steps with tensors

# Vectors, matrices, and tensors

Vector Matrix Tensor



- 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:

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

Then $i \in \mathcal{I}$ 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 $i \in \mathcal{I}$.

# 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![2]

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

[2] Global data storage a few years ago 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 multiplication 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 usual Euclidean geometry and *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.

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

$$\mathrm{vec}(\mathcal{X}) = \begin{bmatrix} x_{111} \\ x_{211} \\ x_{311} \\ x_{121} \\ \vdots \\ \vdots \\ \vdots \\ x_{123} \\ x_{223} \\ x_{323} \end{bmatrix}$$

# Matricization

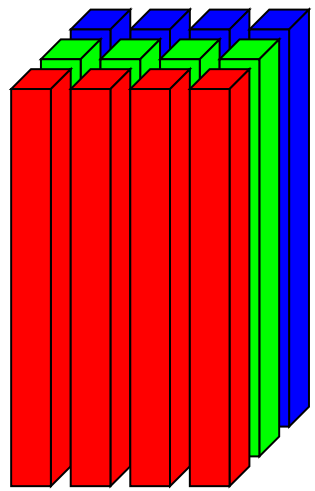
- A matrix has two modes (column mode and row mode).
- A $d$th-order tensor $\mathcal{X}$ has $d$ modes ($\mu = 1$, $\mu = 2$, …, $\mu = d$).

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

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

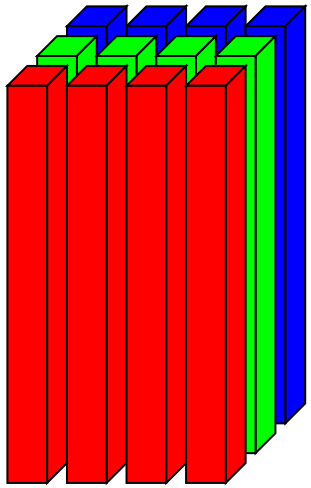
is a vector of length $n_1$ for each choice of $i_2, \ldots, i_d$. These vectors are called fibers.

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

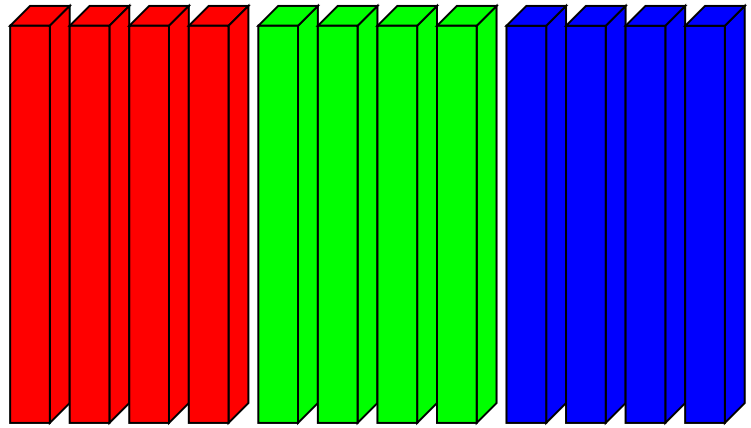


# Matricization

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 $\mu$-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:

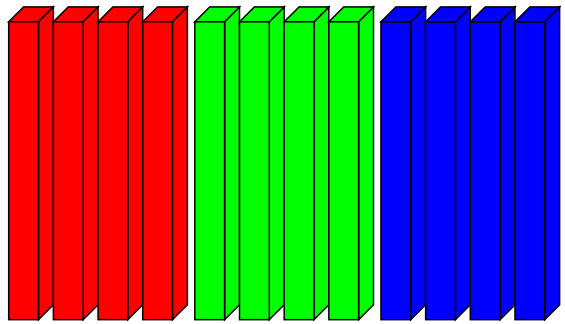
```
b = permute(a,[4 1 2 3]);
reshape(b,5,2*3*4)
```

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

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

# $\mu$-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):

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

# $\mu$-mode matrix products

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

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

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:

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

# $\mu$-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 $\mu$-mode matrix product and matrix-vector product:

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

# 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.
- $\mu$-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 (and all programming languages) offer basic functionality to work with $d$-dimensional arrays.
- MATLAB Tensor Toolbox: `http://www.tensortoolbox.org/`

# Applications of tensors

# 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:

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

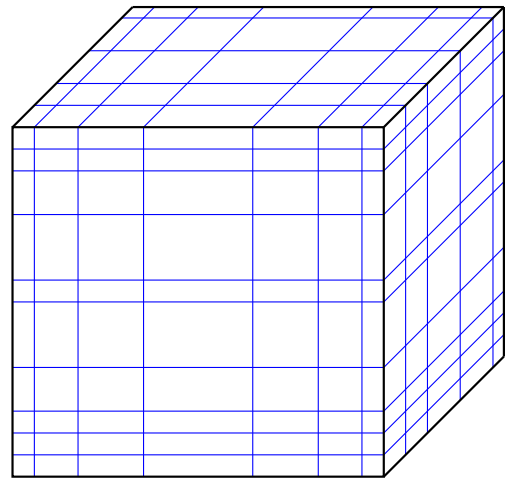
## Typical setting:

- $\mathcal{U}$ 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.

Discretization of function in $d$ variables

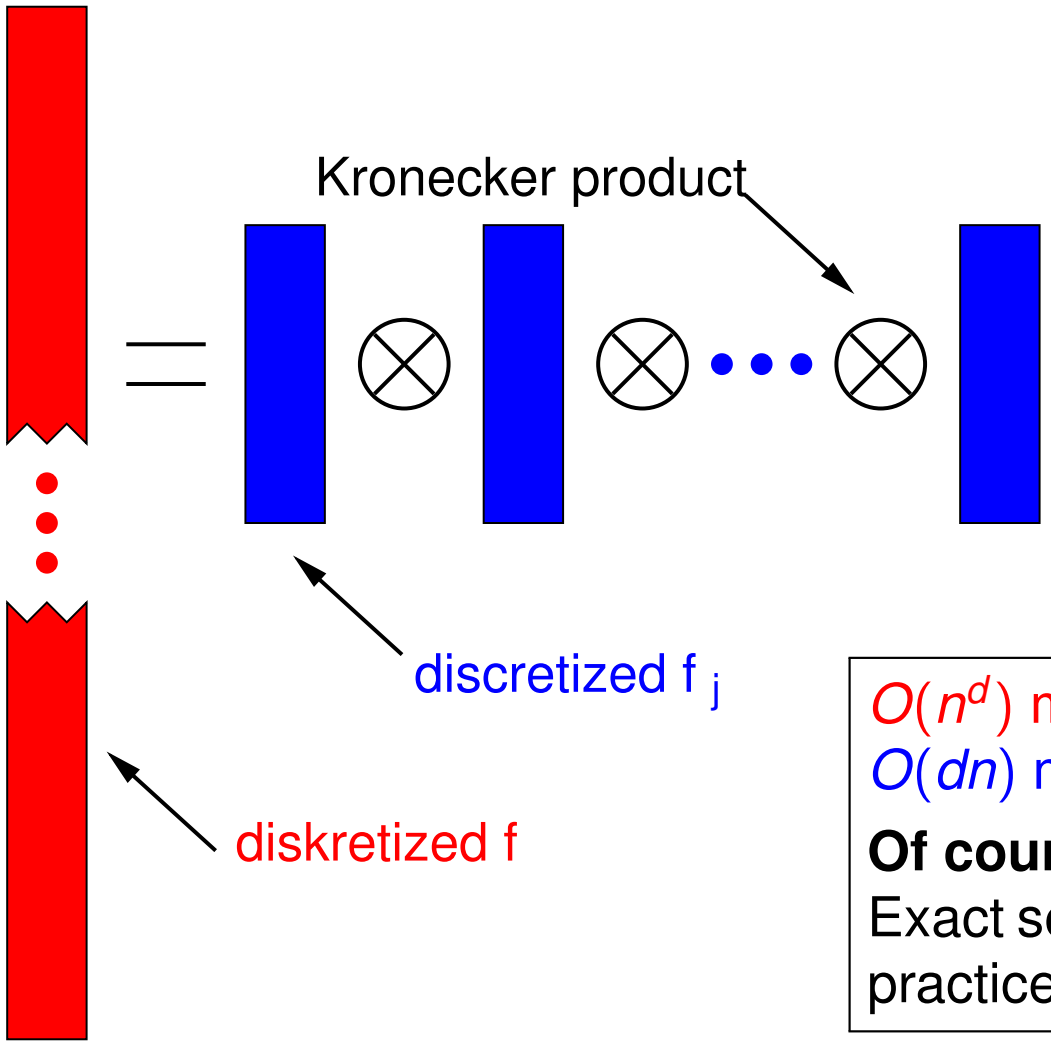
$\xi_1, \ldots, \xi_d \in [0, 1]$

$\rightsquigarrow$ #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 $640 \times 480$ can be viewed as a $640 \times 480 \times 1000$ tensor.

Example 3: Hyperspectral images.

Example 4: Deep learning: Coefficients in each layer of deep NN stored as tensors (TensorFlow), Interpretation of RNNs as hierarchical tensor decomposition.

Typical setting (except for Example 4):

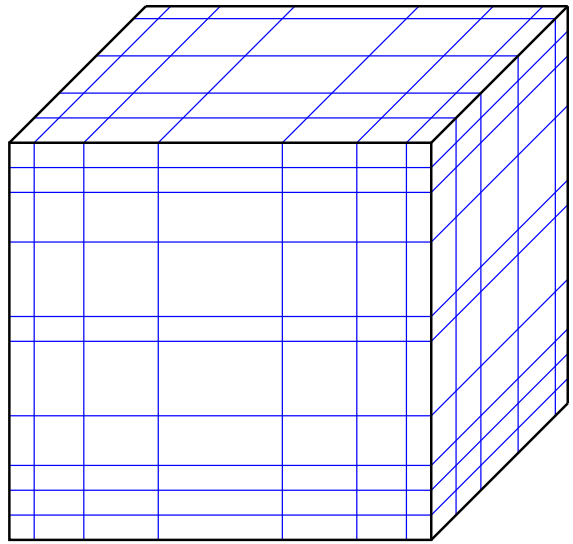
- entries of $\mathcal{U}$ often given explicitly (at least partially).
- extraction of dominant features from $\mathcal{U}$.
- usually moderate values for $d$.

# 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 $\mathcal{U} \in \mathbb{R}^{n \times n \times n}$:

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

# 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:

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

with

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

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

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

# Example: Henon-Heiles potential

Consider $\Omega = [-10, 2]^d$ and potential ([Meyer et al. 1990; Raab et al. 2000; Faou et al. 2009])

$$V(\xi) = \frac{1}{2}\sum_{j=1}^{d} \sigma_j \xi_j^2 + \sum_{j=1}^{d-1} \left( \sigma_* (\xi_j \xi_{j+1}^2 - \frac{1}{3}\xi_j^3) + \frac{\sigma_*^2}{16}(\xi_j^2 + \xi_{j+1}^2)^2 \right).$$

with $\sigma_j \equiv 1$, $\sigma_* = 0.2$.

Discretization with $n = 128$ dof/dimension for $d = 20$ dimensions.

- Eigenvector has $n^d \approx 10^{42}$ entries.
- Explicit storage of eigenvector would require $10^{25}$ exabyte!

Solved with accuracy $10^{-12}$ in less than 1 hour on laptop.

# Quantum many-body problems

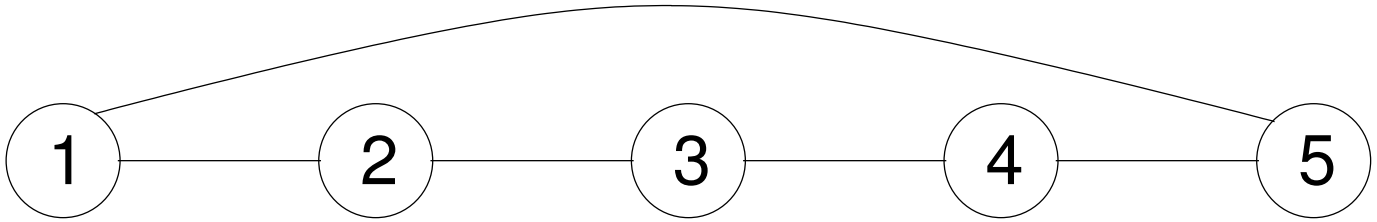
- spin-½ particles: proton, neutron, electron, and quark.
- two states: spin-up, spin-down
- quantum state for each spin represented by vector in $\mathbb{C}^2$ (spinor)
- quantum state for system of $d$ spins represented by vector in $\mathbb{C}^{2^d}$
- quantum mechanical operators expressed in terms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

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

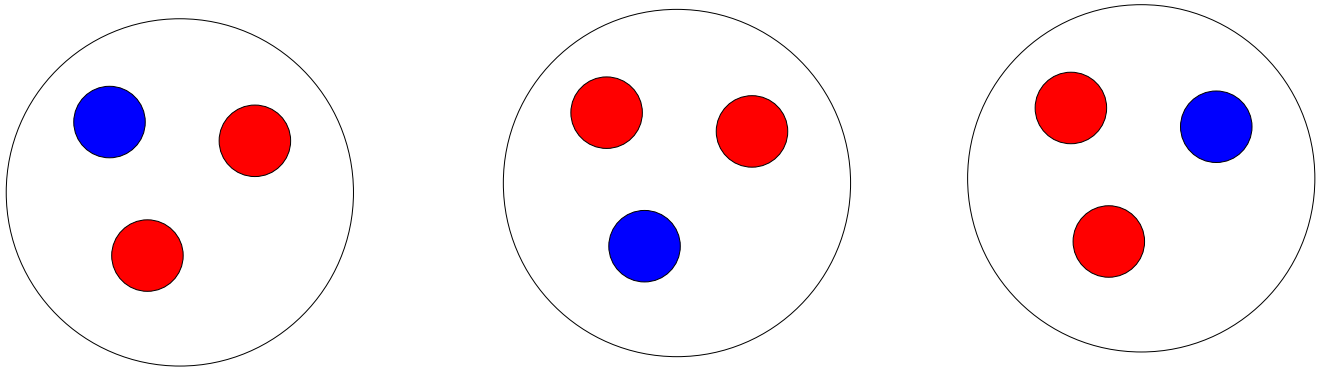
# Quantum many-body problems

- 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)

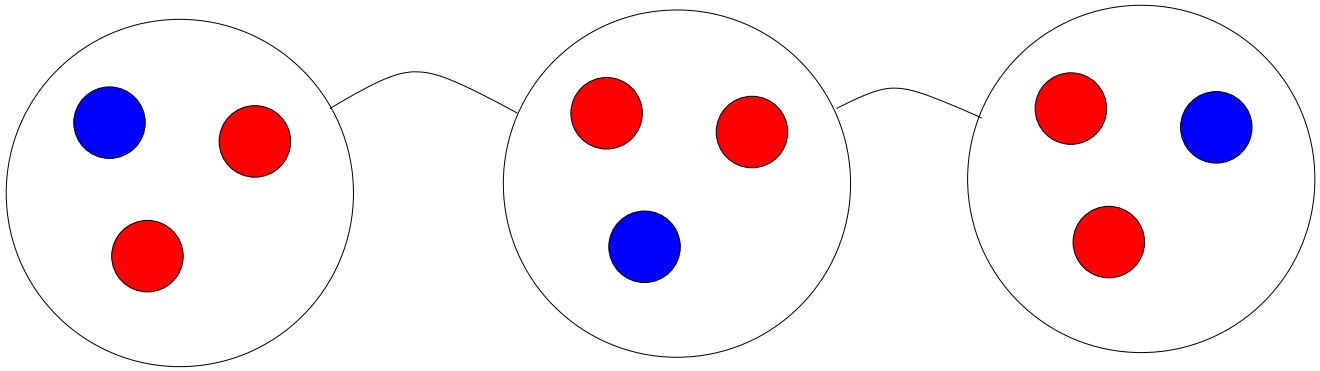


- 3 stochastic automata $A_1, A_2, A_3$ having 3 states each.
- Vector $x_t^{(i)} \in \mathbb{R}^3$ describes probabilities of states (1), (2), (3) in $A_i$ at time $t$
- No coupling between automata $\rightsquigarrow$ local transition $x_t^{(i)} \mapsto x_{t+1}^{(i)}$ described by Markov chain:

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

  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 $\mathcal{X}(t) \in \mathbb{R}^{3\times 3\times 3}$) 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:

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

- 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.

# Low-rank tensor techniques

- Emerged during last five years in numerical analysis.
- Successfully applied to:
  - parameter-dependent / multi-dimensional integrals;
  - electronic structure calculations: Hartree-Fock / DFT;
  - stochastic and parametric PDEs;
  - high-dimensional Boltzmann / chemical master / Fokker-Planck / Schrödinger equations;
  - micromagnetism;
  - rational approximation problems;
  - computational homogenization;
  - computational finance;
  - multivariate regression and machine learning;
  - ...

# References on applications


M. Bachmayr, R. Schneider, and A. Uschmajew.

Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations.
*Found. Comput. Math.*, 16(6):1423–1472, 2016.

A. Cichocki.

Era of big data processing: A new approach via tensor networks and tensor decompositions.
arXiv:1403.2048, 2014.

L. Grasedyck, D. Kressner, and C. Tobler.

A literature survey of low-rank tensor approximation techniques.
*GAMM-Mitt.*, 36(1):53–78, 2013.

W. Hackbusch.

*Tensor Spaces and Numerical Tensor Calculus*.
Springer, Heidelberg, 2012.

V. Khrulkov, A. Novikov, and I. Oseledets.

Expressive power of recurrent neural networks, 2018.
ICLR: Sixth International Conference on Learning Representations.

Anthony Nouy.

Low-rank methods for high-dimensional approximation and model order reduction.
In *Model reduction and approximation*, volume 15 of *Comput. Sci. Eng.*, pages 171–226.
SIAM, Philadelphia, PA, 2017.

N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos.

Tensor decomposition for signal processing and machine learning.
*IEEE Trans. Signal Process.*, 65(13):3551–3582, 2017.