Hands-on advanced numerical methods for quantum many-body dynamics

A tour through numerical methods for simulating large-scale quantum physics (classically)
Motto: Do it from scratch to better understand quantum physics and classical limitations
Structure: Theory lecture part + tutorial-style

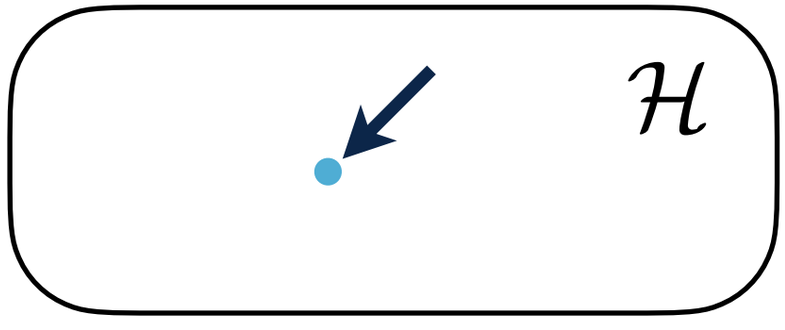
- Language used: Julia, **https://julialang.org/** (open source, easy, fast linear algebra)

- **Outline** (may vary)

1. **20/10:** Basic concepts: Numbers on computers, basic exact diagonalization (ED)
   Tutorial: ED on a simple spin model
2. **21/10:** A better ED, sparse matrices, Krylov space. Open systems.
   Tutorial: Spin-model simulations using Krylov space
3. **22/10**: Mean-field, Runge-Kutta
   Tutorial: A mean-field simulation of the transverse Ising model
4. **23/10:** How to go beyond: Matrix product states

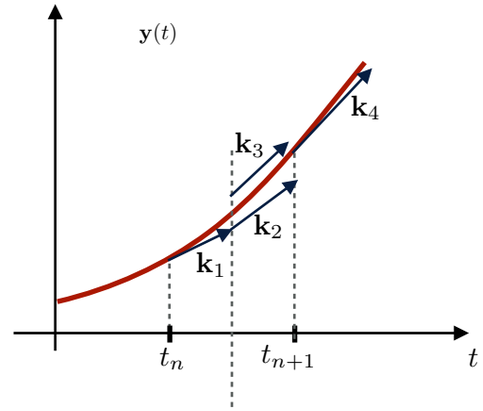
Last time

- Not all quantum problems are linear. Also in quantum mechanics, when making e.g. a mean-field approximation (e.g. a product state approximation for spin models, or in the Gross-Pitaevskii equations for ultra-cold bosonic gases) the **problem becomes non-linear**. However the state-space drastically decreases in this case (from exponential to linear growth with system size). The model is essentially equivalent to "classical spins", entanglement is neglected.

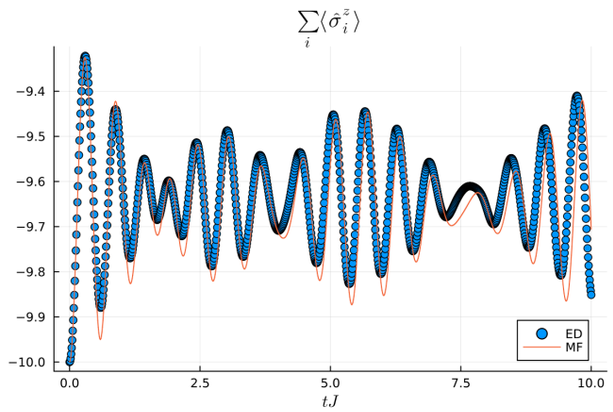


- Runge-Kutta methods are a general tool to simulate dynamics of linear and non-linear problems. Formally derived from Taylor expansions using multiple steps. In particular the 4-th order method is a good compromise (stable, small error for reasonable time-step).

$$
\begin{aligned}
\mathbf{k}_1 &= f(t_n, \mathbf{y}_n) \\
\mathbf{k}_2 &= f\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right) \\
\mathbf{k}_3 &= f\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right) \\
\mathbf{k}_4 &= f\left(t_n + h, \mathbf{y}_n + h\mathbf{k}_3\right) \\
\mathbf{y}_{n+1} &\approx \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)
\end{aligned}
$$



- As example mean-field problem we showed how to simulate dynamics of a transverse Ising model in mean-field and compared to exact results. Generally mean-field works well in low-excitation regimes and for high connectivity

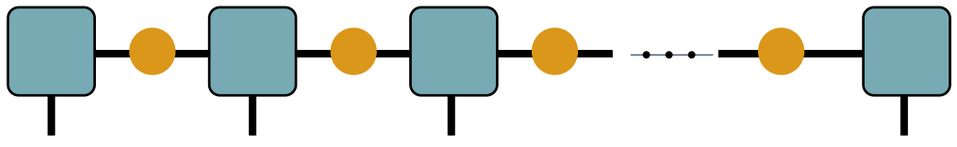


This time

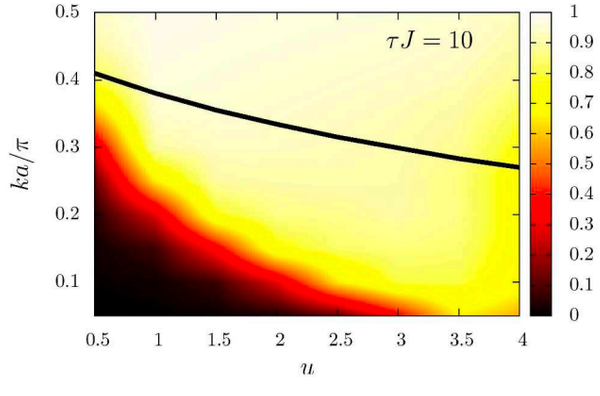
- **Part 1:** Tensors and “tensor-networks”

- **Part 2:** Bi-partite entanglement and bi-partite state compression

- **Part 3:** Matrix product states (MPS) and the time-evolving block decimation algorithm (TEBD) and application examples



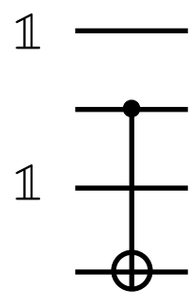
- **Example** of MPS simulations



Tensors

- We have learned how to use tensor products and sparse matrices to construct matrices, e.g. for a unitary gate:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



- Rethinking state-vectors and gates

State = Vector

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \psi_i \, |i\rangle \qquad \begin{bmatrix} \vdots \\ \psi_i \\ \vdots \end{bmatrix} \quad D \times 1$$

$$\begin{aligned} i = 0 \quad &\leftrightarrow \quad |0000\rangle \\ i = 1 \quad &\leftrightarrow \quad |0001\rangle \\ i = 2 \quad &\leftrightarrow \quad |0010\rangle \\ &\vdots \\ i = 16 \quad &\leftrightarrow \quad |1111\rangle \end{aligned}$$

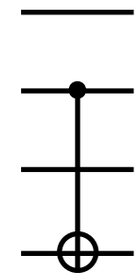
$$i = \sum_{k=1}^{n} i_k 2^{n-k} \quad \leftrightarrow \quad |i_1 i_2 \dots i_n\rangle$$

- We may also think about the state-vector as a n-dimensional state-tensor (n indices)

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \dots i_n} \, |i_1 i_2 \dots i_n\rangle$$

Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$$



- Equivalently the two-qubit gate acting on qubits 2 and 4, can be written as a 4-dimensional tensor:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \text{CNOT} = \sum_{a=0}^{3} \sum_{b=0}^{3} M_a^b |a\rangle \langle b| = \sum_{a_2=0}^{1} \sum_{a_4=0}^{1} \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{a_2 a_4}^{b_2 b_4} |a_2 a_4\rangle \langle b_2 b_4|$$

… implicit meaning: $|a_2 a_4\rangle \langle b_2 b_4| = \mathbb{1} \otimes |a_2\rangle \langle b_2| \otimes \mathbb{1} \otimes |a_4\rangle \langle b_4|$

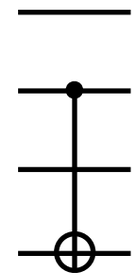
$$\text{CNOT}\,|\psi\rangle = \sum_{a_2=0}^{1} \sum_{a_4=0}^{1} \sum_{i_1=0}^{1} \sum_{i_3=0}^{1} \left(\sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{a_2 a_4}^{b_2 b_4} c_{i_1 b_2 i_3 b_4} \right) |i_1 a_2 i_3 a_4\rangle \equiv \sum_{j_1=0}^{1} \sum_{j_2=0}^{1} \sum_{j_3=0}^{1} \sum_{j_4=0}^{1} \tilde{c}_{j_1 j_2 \ldots j_n} |j_1 j_2 j_3 j_4\rangle$$

- The updated state tensor is obtained from a tensor contraction (summation over joint indices)

$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{i_2 i_4}^{b_2 b_4} c_{i_1 b_2 i_3 b_4}$$

Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} \, |i_1 i_2 \ldots i_n\rangle$$

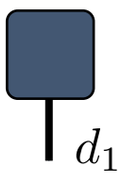


$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M^{b_2 b_4}_{i_2 i_4} c_{i_1 b_2 i_3 b_4}$$

- To make this more intuitive: Introduce a diagrammatic notation

Complex number

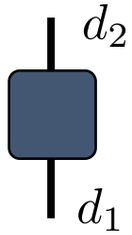
z



Vector

v_i

$i = 1, \ldots, d_1$

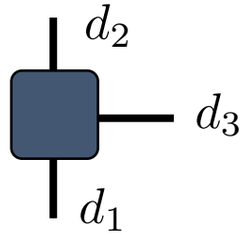


Matrix

M_{ij}

$i = 1, \ldots, d_1$

$j = 1, \ldots, d_2$



3D tensor

$\Gamma_{i,j,k}$

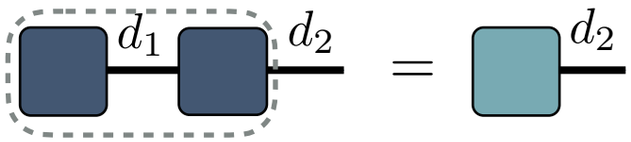
$i = 1, \ldots, d_1$

$j = 1, \ldots, d_2$

Tensors

- **Contractions** of tensors:

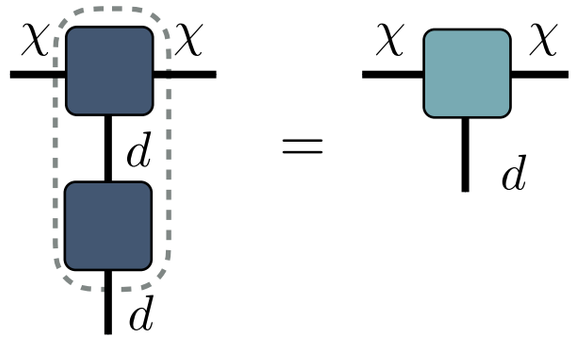
Matrix-Vector product:



$$\sum_{i=1}^{d_1} v_i M_{ij} = \tilde{v}_j$$

$\mathcal{O}(d_1 d_2)$ multiplications

Tensor-matrix contraction:



$\mathcal{O}(d^2\chi^2)$ multiplications

- Useful for avoiding messy indexing in tensor operations|
- **Allows easy estimations of numerical complexity!**

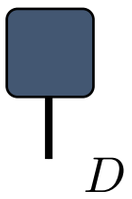
Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} \, |i_1 i_2 \ldots i_n\rangle$$

- **State-vector vs. state-tensor**

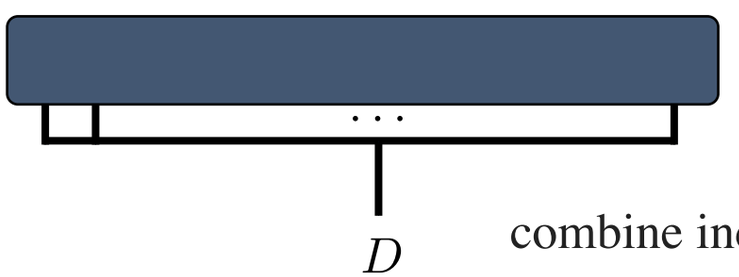
State = Vector

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \psi_i \, |i\rangle \qquad \begin{bmatrix} \vdots \\ \psi_i \\ \vdots \end{bmatrix}$$



$\Leftrightarrow$

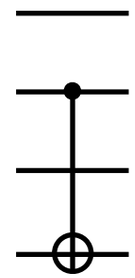
$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} \, |i_1 i_2 \ldots i_n\rangle$$



combine indices $\quad i = \sum_{k=1}^{n} i_k 2^{n-k}$

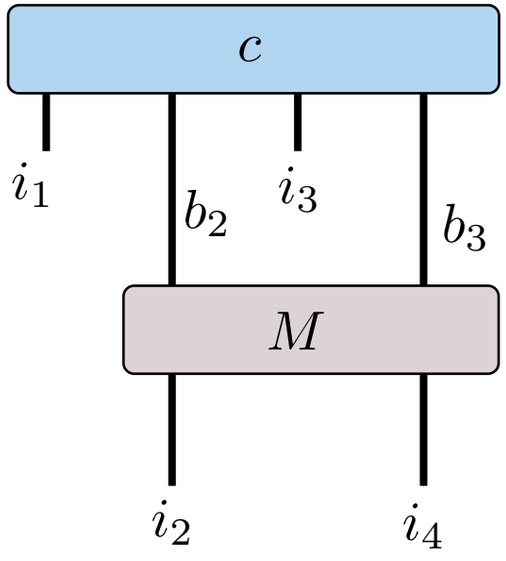
Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} \, |i_1 i_2 \ldots i_n\rangle$$

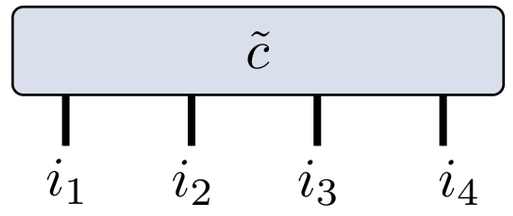


$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M^{b_2 b_4}_{i_2 i_4} \, c_{i_1 b_2 i_3 b_4}$$

- Application of CNOT gate

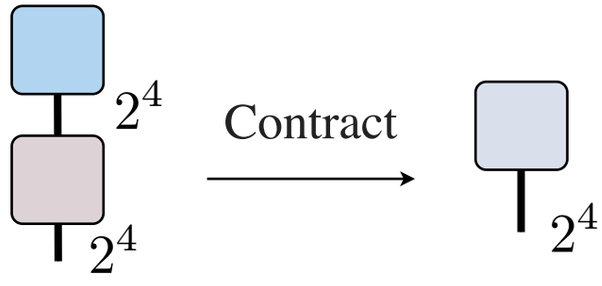


Contract



Cost $\mathcal{O}(2^6)$ multiplications

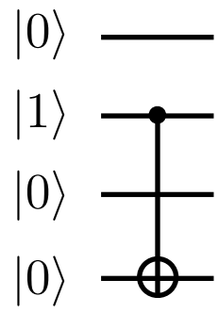
Note: Full matrix-vector multiplication would be $\mathcal{O}(2^8)$



Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$$

$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{i_2 i_4}^{b_2 b_4} c_{i_1 b_2 i_3 b_4}$$



- Contraction in code:

```
# define index 1 = |0>, index 2 = |1>
dims = (2, 2, 2, 2)

                          "multi-dimensional array"
c = zeros(T, dims)
c[1, 2, 1, 1] = 1.0
# state |0 1 0 0>
# ... should go to |0 1 0 1>

tc = Array{T}(undef, dims) # only allocated

# define order i2, i4, b2, b4
M = zeros(T, dims)
M[1, 1, 1, 1] = 1.0 # |00> -> |00>
M[1, 2, 1, 2] = 1.0 # |01> -> |01>
M[2, 2, 2, 1] = 1.0 # |10> -> |11>
M[2, 1, 2, 2] = 1.0 # |11> -> |10>
```

$$\mathrm{CNOT}\,|00\rangle = |00\rangle$$

$$\mathrm{CNOT}\,|01\rangle = |01\rangle$$

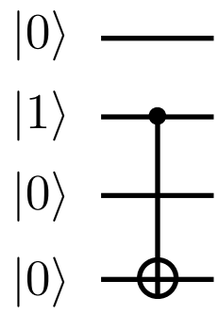
$$\mathrm{CNOT}\,|10\rangle = |11\rangle$$

$$\mathrm{CNOT}\,|11\rangle = |10\rangle$$

Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$$

$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{i_2 i_4}^{b_2 b_4} c_{i_1 b_2 i_3 b_4}$$



- Three ways of doing the contraction in code
- **I. Loops with multi-dimensional arrays**

```
# define index 1 = |0>, index 2 = |1>
dims = (2, 2, 2, 2)

c = zeros(T, dims)
c[1, 2, 1, 1] = 1.0
# state |0 1 0 0>
# ... should go to |0 1 0 1>

tc = Array{T}(undef, dims) # only allocated
```

```
# define order i2, i4, b2, b4
M = zeros(T, dims)
M[1, 1, 1, 1] = 1.0 # |00> -> |00>
M[1, 2, 1, 2] = 1.0 # |01> -> |01>
M[2, 2, 2, 1] = 1.0 # |10> -> |11>
M[2, 1, 2, 2] = 1.0 # |11> -> |10>
```

```
# apply CNOT gate to indices 2 and 4
for i1 = 1:2
    for i2 = 1:2
        for i3 = 1:2
            for i4 = 1:2

                tc[i1, i2, i3, i4] = 0.0

                for b2 = 1:2
                    for b4 = 1:2
                        tc[i1, i2, i3, i4] += M[i2, i4, b2, b4] * c[i1, b2, i3, b4]
                    end
                end
            end
        end
    end
end
```

Cost $\mathcal{O}(2^6)$

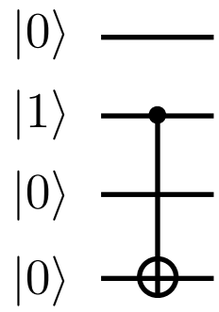
multiplications,
very obvious

Contra:
Cumbersome to code

Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$$

$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M_{i_2 i_4}^{b_2 b_4} c_{i_1 b_2 i_3 b_4}$$



- Three ways of doing the contraction in code
- **II. Fallback to matrix-matrix multiplication routines**

$$\sum_{c_1,c_2} A_{x_1 x_2 c_1 c_2} B_{c_1 c_2 y_1}$$ … this is a matrix multiplication!

```
# apply CNOT gate to indices 2 and 4 using matrix mutliplications

# order in c: i1, b2, i3, b4
c_perm = permutedims(c, (2, 4, 1, 3)) # new order: b2, b4, i1, i3

c_perm_mat = reshape(c_perm, 4, 4)
M_mat = reshape(M, 4, 4) # order (i2, i4), (b2, b4)

tc_perm_mat = M_mat * c_perm_mat # now order: i2, i4, i1, i3

tc_perm = reshape(tc_perm_mat, (2, 2, 2, 2))
tc = permutedims(tc_perm, (3, 1, 4, 2))
```

```
# define index 1 = |0>, index 2 = |1>
dims = (2, 2, 2, 2)

c = zeros(T, dims)
c[1, 2, 1, 1] = 1.0
# state |0 1 0 0>
# ... should go to |0 1 0 1>

tc = Array{T}(undef, dims) # only allocated

# define order i2, i4, b2, b4
M = zeros(T, dims)
M[1, 1, 1, 1] = 1.0 # |00> -> |00>
M[1, 2, 1, 2] = 1.0 # |01> -> |01>
M[2, 2, 2, 1] = 1.0 # |10> -> |11>
M[2, 1, 2, 2] = 1.0 # |11> -> |10>
```

Pro:

More compact

Built-in matrix multiplications can be very fast

(BLAS routines, e.g. INTEL MKL)

Contra:

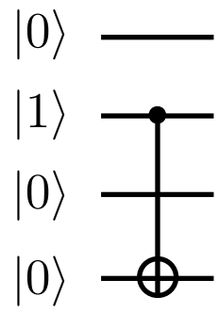
Easy to make mistakes

Permutations add memory allocations

Tensors

$$|\psi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle$$

$$\tilde{c}_{i_1 i_2 i_3 i_4} = \sum_{b_2=0}^{1} \sum_{b_4=0}^{1} M^{b_2 b_4}_{i_2 i_4} c_{i_1 b_2 i_3 b_4}$$



- Three ways of doing the contraction in code
- **II. Fallback to matrix-matrix multiplication routines**

… and using a package doing it for you.

```julia
# apply CNOT gate to indices 2 and 4 using TensorOperations
using TensorOperations
@tensor tc[i1, i2, i3, i4] = M[i2, i4, b2, b4] * c[i1, b2, i3, b4]
```

(nice package, can optimize multi-tensor-contraction ordering at compile time)

https://jutho.github.io/TensorOperations.jl

```julia
# define index 1 = |0>, index 2 = |1>
dims = (2, 2, 2, 2)

c = zeros(T, dims)
c[1, 2, 1, 1] = 1.0
# state |0 1 0 0>
# ... should go to |0 1 0 1>

tc = Array{T}(undef, dims) # only allocated

# define order i2, i4, b2, b4
M = zeros(T, dims)
M[1, 1, 1, 1] = 1.0 # |00> -> |00>
M[1, 2, 1, 2] = 1.0 # |01> -> |01>
M[2, 2, 2, 1] = 1.0 # |10> -> |11>
M[2, 1, 2, 2] = 1.0 # |11> -> |10>
```

Pro:
More compact
Built-in matrix multiplications can be very fast

(BLAS routines, e.g. INTEL MKL)

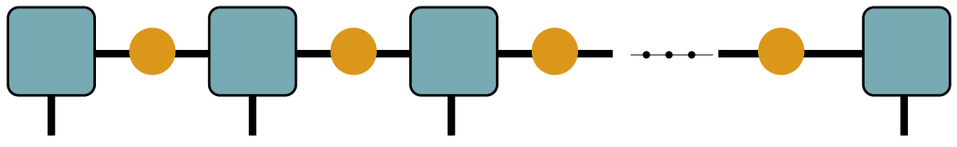
Contra:
Still can add unnecessary overhead

This time

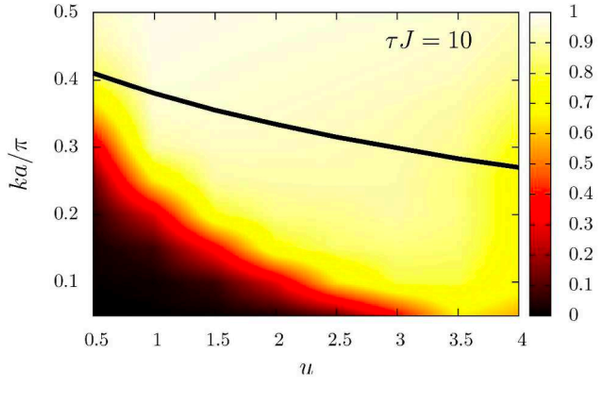
- **Part 1:** Tensors and “tensor-networks”

- **Part 2:** Bi-partite entanglement and bi-partite state compression

- **Part 3:** Matrix product states (MPS) and the time-evolving block decimation algorithm (TEBD) and application examples

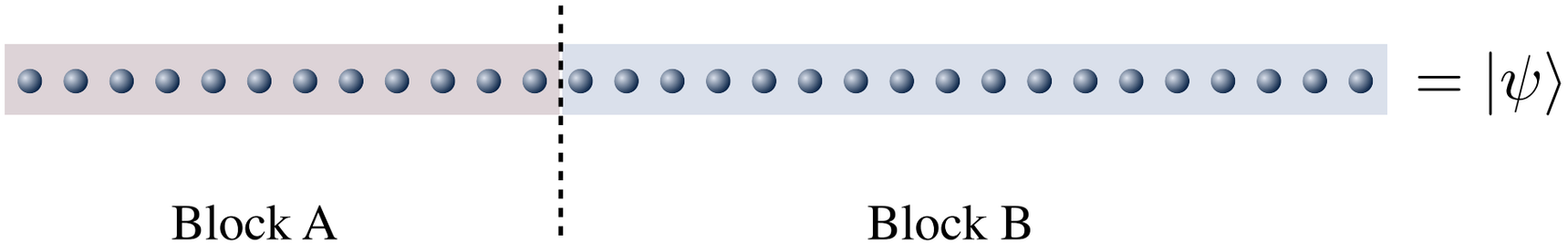


- **Examples:** MPS simulations



Bipartite entanglement and state compression

- Assume a pure state of our many-body system (qubits/spins or bosons, …)



- Bipartition:
- Definition: $|\psi\rangle \neq |\phi\rangle_A |\chi\rangle_B$ ⟷ "Blocks A and B are entangled"
- How to quantify the amount of entanglement?

State in block A

$$\hat{\rho}_A = \mathrm{tr}_B \left(|\psi\rangle\langle\psi| \right)$$

State in block B

$$\hat{\rho}_B = \mathrm{tr}_A \left(|\psi\rangle\langle\psi| \right)$$

- **Example:** Say the total state is a non-entangled product state $|\psi\rangle = |\phi\rangle_A |\chi\rangle_B$

… then:

$$\hat{\rho}_A = |\phi\rangle \langle\phi|_A \qquad \hat{\rho}_B = |\chi\rangle \langle\chi|_B$$

Pure states!

The state of each block contains all information of the state in that block!

Bipartite entanglement and state compression

- Assume a pure state of our many-body system (qubits/spins or bosons, …)

- Bipartition:
- Definition: $|\psi\rangle \neq |\phi\rangle_A \, |\chi\rangle_B$ ⟷ "Blocks A and B are entangled"

- **Example:** Say the total state of just two qubits is maximally entangled: $|\psi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \, |0\rangle_B + |1\rangle_A \, |1\rangle_B \right)$

… then: $\hat{\rho}_A = \mathrm{tr}_B \left(|\psi\rangle\langle\psi| \right) = \langle 0|\psi\rangle\langle\psi|0\rangle_B + \langle 1|\psi\rangle\langle\psi|1\rangle_B$

$$|\psi\rangle\langle\psi| = \frac{1}{2} \left(|0\rangle_A |0\rangle_B \langle 0|_A \langle 0|_B + |0\rangle_A |0\rangle_B \langle 1|_A \langle 1|_B + |1\rangle_A |1\rangle_B \langle 0|_A \langle 0|_B + |1\rangle_A |1\rangle_B \langle 1|_A \langle 1|_B \right)$$

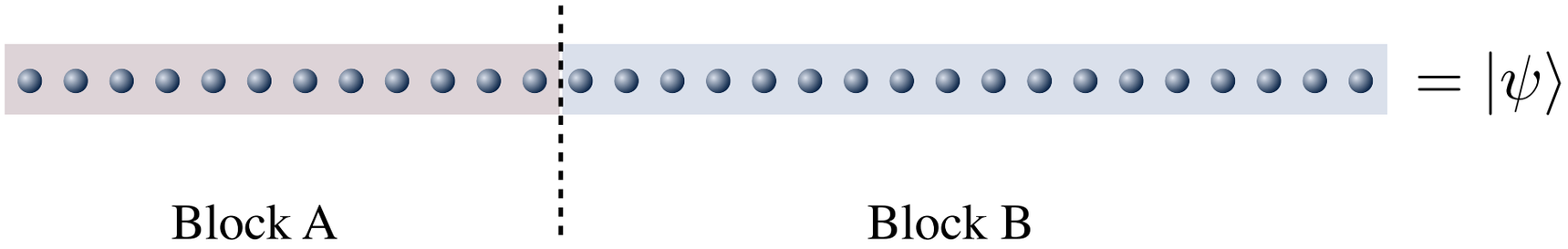
$$\hat{\rho}_A = \frac{1}{2} \left(|0\rangle\langle 0|_A + |1\rangle\langle 1|_A \right) = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}_A$$

The state of a sub-block is a maximally mixed state!

Idea: Use the **entropy** of the reduced density matrix as entanglement measure!

Bipartite entanglement and state compression

- Assume a pure state of our many-body system



- Bipartition:
- Definition: $|\psi\rangle \neq |\phi\rangle_A \, |\chi\rangle_B$ ⟷ "Blocks A and B are entangled"
- How to quantify the amount of entanglement?

State in block A

$$\rho_A = \mathrm{tr}_B \left(|\psi\rangle\langle\psi| \right)$$

State in block B

$$\rho_B = \mathrm{tr}_A \left(|\psi\rangle\langle\psi| \right)$$

- **von Neumann entanglement entropy:**

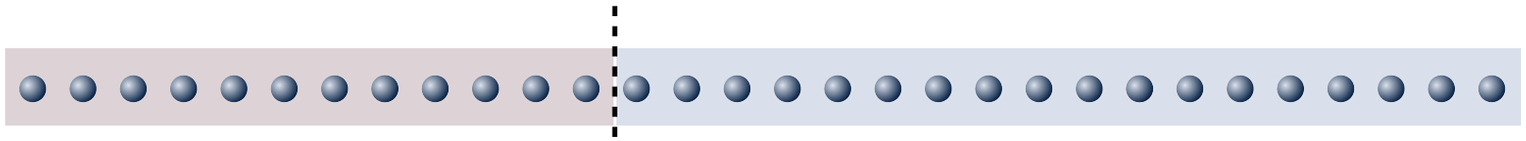
true for pure states on the full system

$$S_{\mathrm{vN}}(\hat{\rho}_A) = -\mathrm{tr}_{\mathrm{A}} \left[\hat{\rho}_A \log_2(\hat{\rho}_A) \right] = -\sum_\alpha \lambda_\alpha \log_2(\lambda_\alpha) = S_{\mathrm{vN}}(\rho_B)$$

Eigenvalues of ρ_A

Bipartite entanglement and state compression

- Assume a pure state of our many-body system



$$= |\psi\rangle$$

- Bipartition:

Block A

$d_A = \dim(\mathcal{H}_A)$

Block B

$w.l.o.g.\ d_A < \dim(\mathcal{H}_B) = d_B$

- Entanglement entropy:

$$S_{\mathrm{vN}}(\hat{\rho}_A) = -\mathrm{tr}_{\mathrm{A}}\left[\hat{\rho}_A \log_2(\hat{\rho}_A)\right] = -\sum_\alpha \lambda_\alpha \log_2(\lambda_\alpha) = S_{\mathrm{vN}}(\rho_B)$$

No entanglement — **Maximum** entanglement

$\lambda_1 = 1, \lambda_{\alpha\neq 1} = 0 \quad S_{\mathrm{vN}}(\rho_A) = 0$

$\lambda_\alpha = \dfrac{1}{d_A} \quad S_{\mathrm{vN}}(\rho_A) = \log_2(d_A)$

- Two qubit examples:

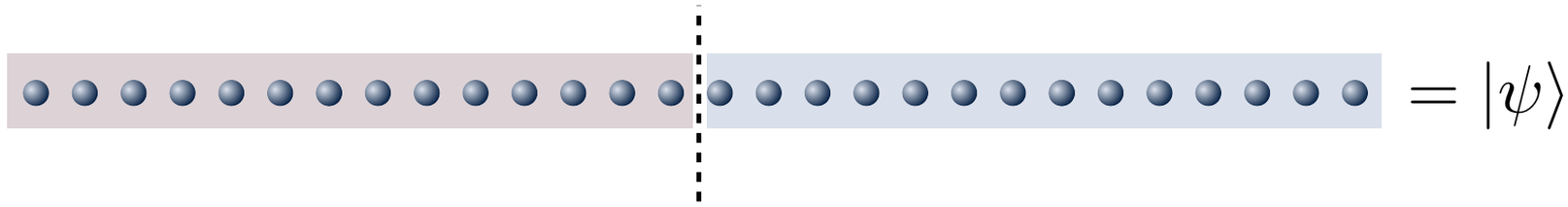
$$|\psi\rangle = |0\rangle_A |1\rangle_B \leftrightarrow \rho_A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}_A$$

$$|\psi\rangle = \frac{1}{2}\left(|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B\right) \leftrightarrow \rho_A = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}_A$$

$$S(\hat{\rho}_A) = -1 \log_2(1) = 0$$

$$S(\hat{\rho}_A) = -2\frac{1}{2}\log_2\left(\frac{1}{2}\right) = \log_2(2) = 1$$

Bipartite entanglement and state compression



$= |\psi\rangle$

- General state in some basis: $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j} |i\rangle_A |j\rangle_B$ $\qquad (\mathbf{C})_{i,j} = c_{i,j}$

- **Singular value decomposition (SVD)** of the coefficient matrix (also known as **Schmidt decomposition**)

$$c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j} \qquad \mathbf{C} = \mathbf{USV}^\dagger$$

U,**V** unitary, **S**: real & diagonal

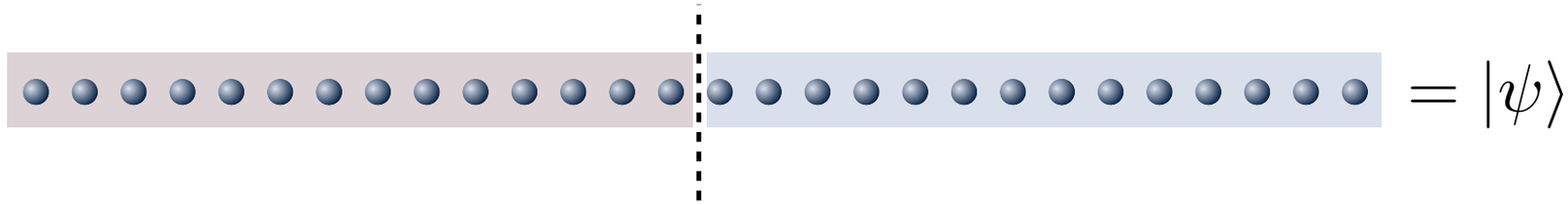
… linear algebra textbook

“Singular values”

- Now we can compute the entropy of a subsystem after the SVD

$$\hat{\rho} = \sum_{i',j'} c^*_{i',j'} \sum_{i,j} c_{i,j} |ij\rangle\langle i'j'| \qquad \hat{\rho}_A = \sum_n \langle n|\rho|n\rangle_2 \qquad \hat{\rho}_A = \sum_{i',i'} \sum_n c^*_{i',n} c_{i,n} |i\rangle\langle i'|$$

Bipartite entanglement and state compression



$= |\psi\rangle$

Singular value decomposition (SVD) $\quad d_A = d_B = d$

$$|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j} |i\rangle_A |j\rangle_B \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$$

U,**V** unitary, **S**: real & diagonal

$$(\mathbf{C})_{i,j} = c_{i,j} \quad \mathbf{C} = \mathbf{USV}^\dagger$$

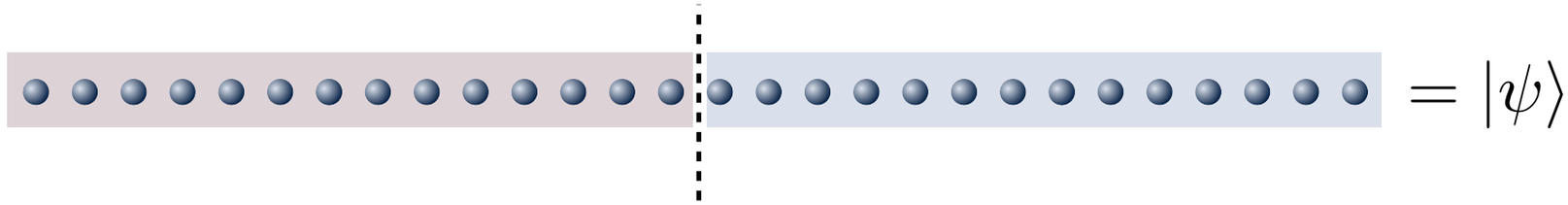
- Now we can compute the entropy of a subsystem after the SVD

$$\hat{\rho} = \sum_{i',j'} c^*_{i',j'} \sum_{i,j} c_{i,j} |ij\rangle\langle i'j'| \qquad \hat{\rho}_A = \sum_n \langle n|\rho|n\rangle_2 \qquad \hat{\rho}_A = \sum_{i',i'} \sum_n c^*_{i',n} c_{i,n} |i\rangle\langle i'|$$

• • • (Exercise)

$$\hat{\rho}_A = \sum_{\alpha} (s_{\alpha,\alpha})^2 |\alpha\rangle\langle\alpha|$$

Bipartite entanglement and state compression



$= |\psi\rangle$

$$d_A = d_B = d$$

Singular value decomposition (SVD)

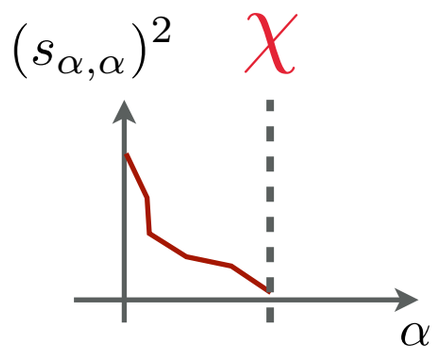
$$|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j} |i\rangle_A |j\rangle_B \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$$

U,**V** unitary, **S**: real & diagonal

$$(\mathbf{C})_{i,j} = c_{i,j} \quad \mathbf{C} = \mathbf{USV}^\dagger$$

$$\hat{\rho}_A = \sum_{\alpha} (s_{\alpha,\alpha})^2 |\alpha\rangle\langle\alpha|$$

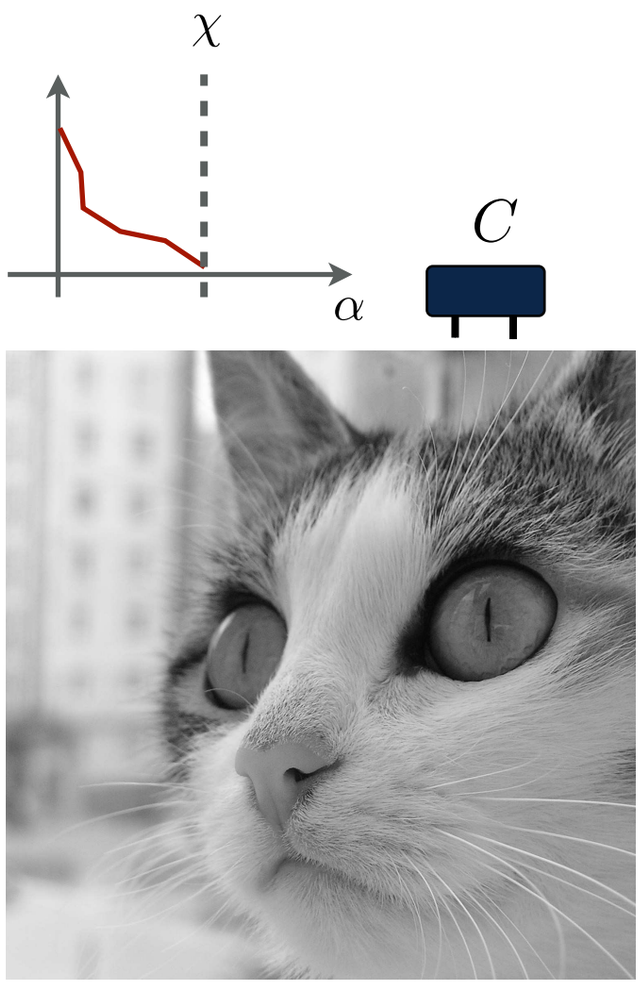
- The SVD diagonalizes the reduced density matrix $\quad S_{\mathrm{vN}}(\hat{\rho}_A) = -\sum_{\alpha}^{d} (s_{\alpha,\alpha})^2 \log_2[(s_{\alpha,\alpha})^2]$
- Singular (Schmidt values) are the square root of the eigenvalues of the reduced density matrix
- Upper limit of entanglement entropy: $\max[S_{\mathrm{vN}}(\hat{\rho}_A)] = \log_2(d)$
- **We can now make an approximation:** What if we just keep the largest singular values?



- Then $\max[S_{\mathrm{vN}}(\rho_A)] = \log_2(\chi)$ … and we now have a state approximation with limited bipartite entanglement
- **Remark:** This is the origin of the naming of an algorithm denoted as “density matrix renormalization group” (DMRG)

Bipartite entanglement and state compression

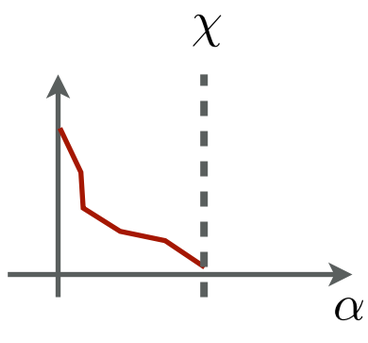
- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let's look at the singular values:




```
using FileIO
using LinearAlgebra
using Plots
using LaTeXStrings

let

C = Float64.(load("cat.png"))

U, S, V = svd(C)

p = plot(log10.(S.^2),
    xlabel = L"\alpha", ylabel = L"\log(s_{\alpha,\alpha}^2)",
    legend = false)
display(p)

end
```

Bipartite entanglement and state compression

- Example for a bipartite “quantum system” $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let’s look at the singular values:

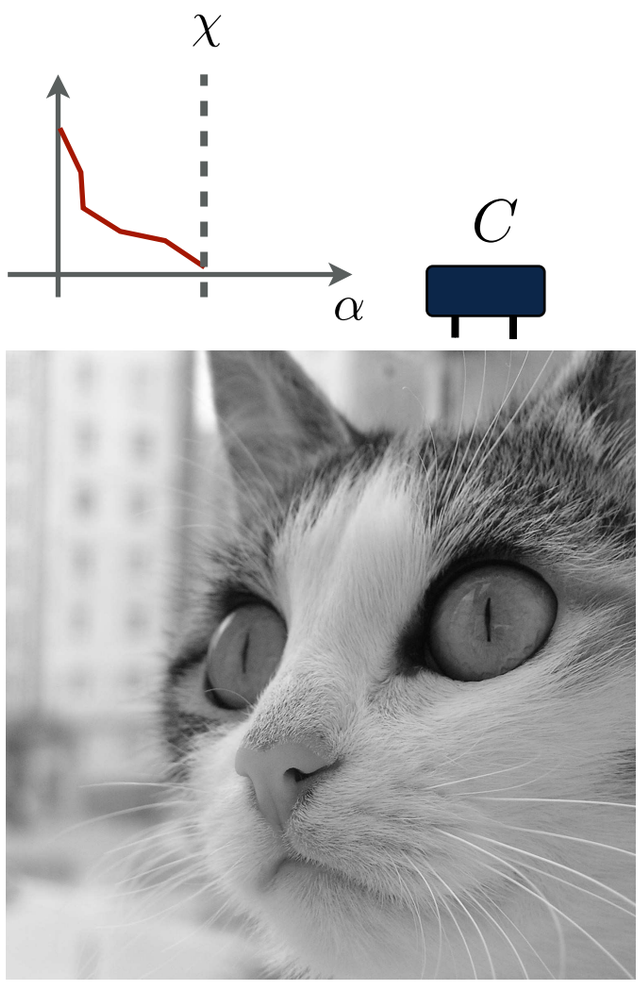


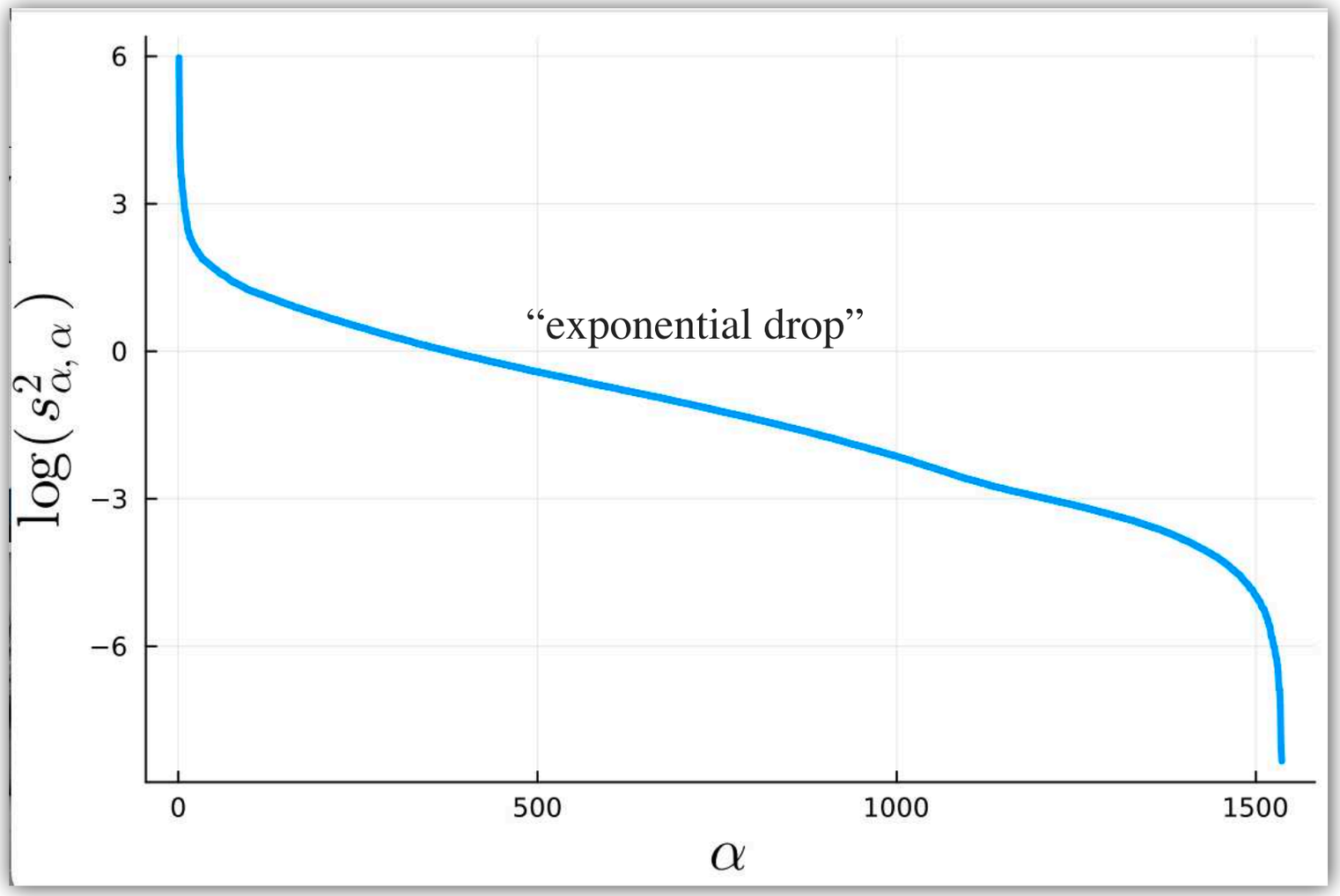
$d = 1536$

```
julia> C
1536×1536 Matrix{Float64}:
 0.909804  0.901961  0.894118  0.890196  0.894118  0.901961  0.909804  0.913725  …  0.462745  0.462745  0.45098   0.447059  0.454902  0.462745  0.466667
 0.909804  0.901961  0.898039  0.894118  0.901961  0.909804  0.913725  0.917647     0.45098   0.458824  0.458824  0.454902  0.45098   0.439216  0.423529
 0.905882  0.901961  0.898039  0.898039  0.901961  0.909804  0.909804  0.909804     0.447059  0.454902  0.454902  0.454902  0.45098   0.427451  0.407843
 0.901961  0.898039  0.894118  0.898039  0.898039  0.901961  0.901961  0.898039     0.454902  0.45098   0.447059  0.447059  0.45098   0.447059  0.439216
 0.901961  0.901961  0.898039  0.898039  0.898039  0.898039  0.894118  0.890196     0.454902  0.45098   0.443137  0.443137  0.458824  0.470588  0.478431
 0.909804  0.901961  0.898039  0.898039  0.898039  0.898039  0.898039  0.894118  …  0.439216  0.447059  0.447059  0.45098   0.466667  0.478431  0.486275
 0.905882  0.898039  0.894118  0.894118  0.898039  0.898039  0.898039  0.898039     0.427451  0.439216  0.45098   0.458824  0.470588  0.47451   0.470588
 0.898039  0.890196  0.886275  0.886275  0.890196  0.898039  0.901961  0.901961     0.423529  0.435294  0.447059  0.462745  0.47451   0.470588  0.458824
 0.898039  0.890196  0.882353  0.878431  0.882353  0.886275  0.886275  0.886275     0.431373  0.439216  0.454902  0.458824  0.454902  0.458824  0.470588
 0.898039  0.890196  0.882353  0.878431  0.882353  0.886275  0.886275  0.886275     0.439216  0.47451   0.482353  0.462745  0.439216  0.447059  0.470588
 0.898039  0.890196  0.882353  0.878431  0.878431  0.882353  0.886275  0.886275  …  0.443137  0.478431  0.486275  0.466667  0.447059  0.454902  0.47451
 0.894118  0.886275  0.878431  0.87451   0.87451   0.878431  0.882353  0.882353     0.427451  0.435294  0.447059  0.454902  0.458824  0.466667  0.47451
 ⋮                                                 ⋮                    ⋱           ⋮                                                   ⋮
 0.87451   0.878431  0.878431  0.87451   0.870588  0.870588  0.87451   0.878431     0.6       0.588235  0.564706  0.560784  0.576471  0.596078  0.603922
 0.87451   0.878431  0.878431  0.878431  0.87451   0.87451   0.87451   0.87451   …  0.576471  0.588235  0.592157  0.588235  0.588235  0.592157  0.6
 0.870588  0.87451   0.882353  0.882353  0.878431  0.87451   0.870588  0.870588     0.580392  0.6       0.619608  0.619608  0.596078  0.588235  0.596078
 0.866667  0.87451   0.882353  0.886275  0.882353  0.87451   0.870588  0.870588     0.603922  0.592157  0.6       0.592157  0.576471  0.588235  0.615686
 0.882353  0.878431  0.87451   0.878431  0.878431  0.878431  0.87451   0.870588     0.611765  0.592157  0.592157  0.6       0.596078  0.588235  0.588235
 0.882353  0.878431  0.87451   0.878431  0.882353  0.882353  0.878431  0.87451      0.639216  0.619608  0.611765  0.603922  0.592157  0.584314  0.588235
 0.878431  0.878431  0.878431  0.878431  0.882353  0.882353  0.878431  0.878431  …  0.662745  0.643137  0.627451  0.607843  0.592157  0.584314  0.588235
 0.882353  0.878431  0.878431  0.878431  0.878431  0.878431  0.87451   0.87451      0.678431  0.654902  0.631373  0.611765  0.592157  0.584314  0.584314
 0.882353  0.882353  0.878431  0.878431  0.87451   0.870588  0.866667  0.866667     0.701961  0.666667  0.639216  0.619608  0.603922  0.592157  0.580392
 0.878431  0.878431  0.878431  0.878431  0.870588  0.866667  0.866667  0.866667     0.717647  0.690196  0.658824  0.635294  0.623529  0.603922  0.584314
 0.87451   0.878431  0.878431  0.878431  0.87451   0.870588  0.870588  0.87451      0.717647  0.701961  0.678431  0.654902  0.639216  0.619608  0.6
 0.866667  0.87451   0.878431  0.878431  0.87451   0.87451   0.878431  0.882353  …  0.701961  0.705882  0.690196  0.670588  0.65098   0.631373  0.615686
```

Bipartite entanglement and state compression

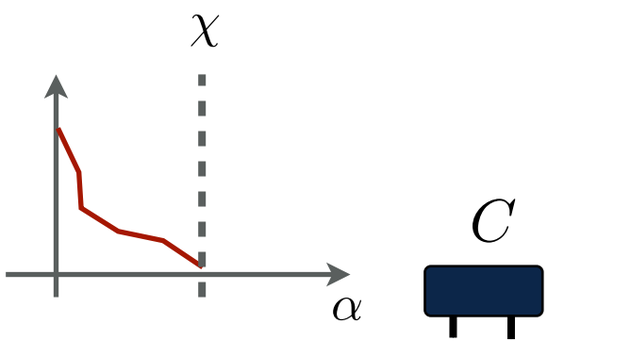
- Example for a bipartite “quantum system” $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let’s look at the singular values:





Bipartite entanglement and state compression

- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let's look at the singular values:



Let's keep only the larges singular value, enforce $\chi = 1$

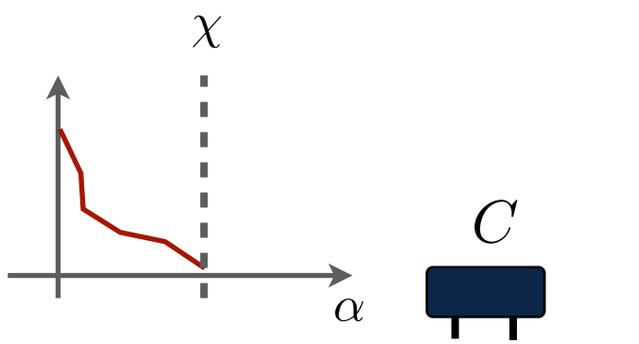
```
let

C = Float64.(load("cat.png"))

U, S, V = svd(C)

chi = 1

@views C_cut = U[:, 1:chi] * Diagonal(S[1:chi]) * V'[1:chi, :]

p = plot(Gray.(C_cut))
display(p)

end
```

Bipartite entanglement and state compression

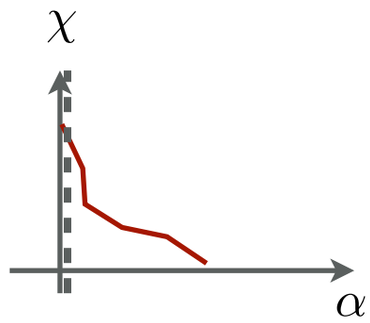
- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$

- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:

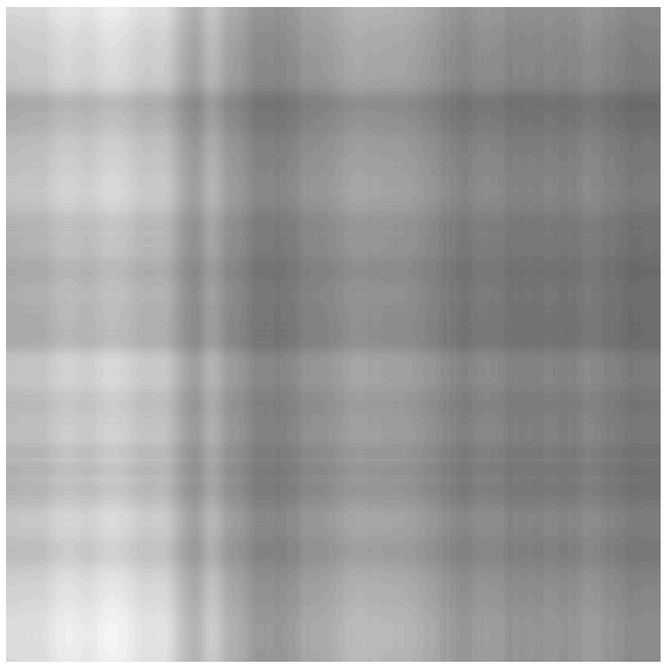
- Doing a SVD of this matrix, let's look at the singular values:



Let's keep only the largest singular value, enforce $\chi = 1 \quad S_{\mathrm{vN}} \leq 0$



$S_{\mathrm{vN}}(\rho_A) = 0$

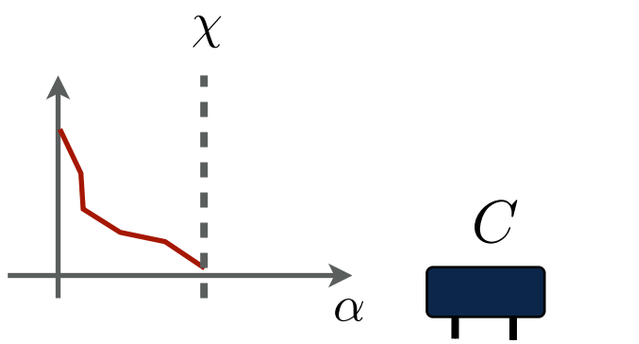


Product state cat
(mean-field cat)

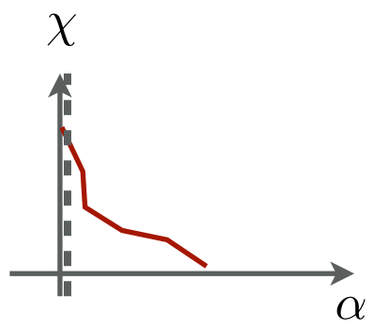
Bad approximation

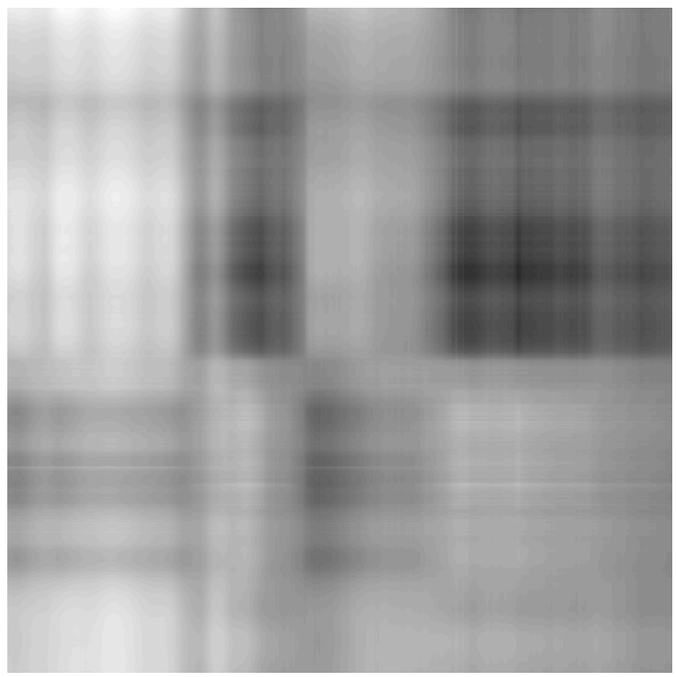
Bipartite entanglement and state compression

- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let's look at the singular values:



Let's keep only the largest few singular values $\chi = 2 \quad S_{\text{vN}} \leq 1$



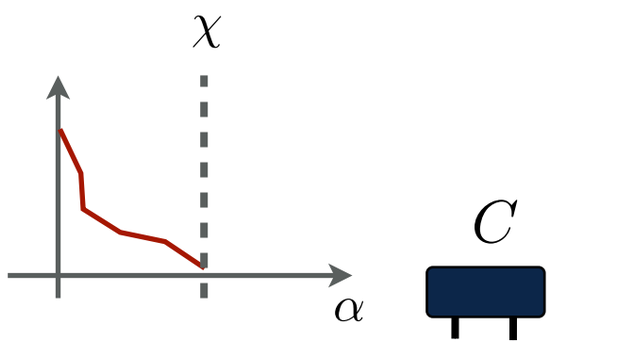


Almost product state cat

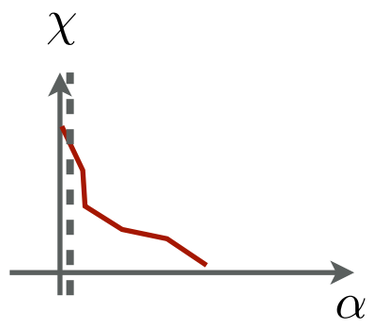
Bad approximation

Bipartite entanglement and state compression

- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$
- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:
- Doing a SVD of this matrix, let's look at the singular values:



Let's keep only the largest few singular values $\chi = 16 \quad S_{\mathrm{vN}} \leq 4$



slightly entangled cat state cat

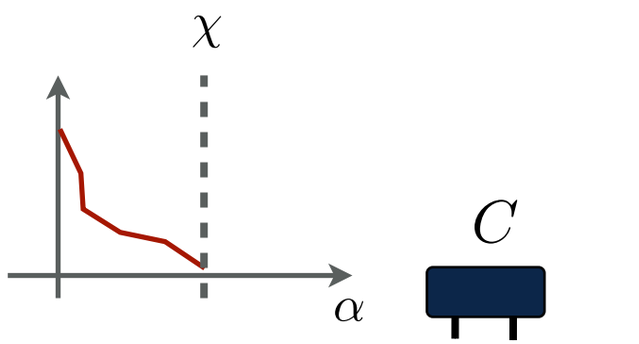
Ok approximation

Bipartite entanglement and state compression

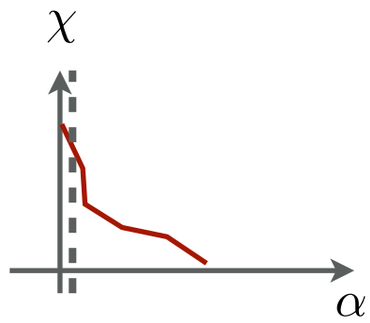
- Example for a bipartite "quantum system" $|\psi\rangle = \sum_{i,j=1}^{d} c_{i,j}|ij\rangle \qquad c_{i,j} = \sum_{\alpha}^{d} u_{i,\alpha} s_{\alpha,\alpha} v^*_{\alpha,j}$

- The matrix C describes all properties of the bi-partite quantum system, i.e. it can be a matrix with numbers describing a gray-scale value of a picture, e.g. a cat:

- Doing a SVD of this matrix, let's look at the singular values:



Let's keep only the largest few singular values $\chi = 128 \quad S_{\mathrm{vN}} \leq 7$



$\chi \ll 1536$

Slightly entangled cat

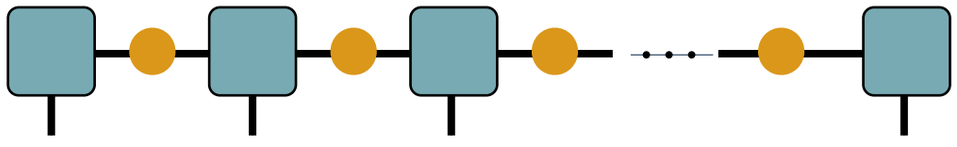
Excellent approximation

This time

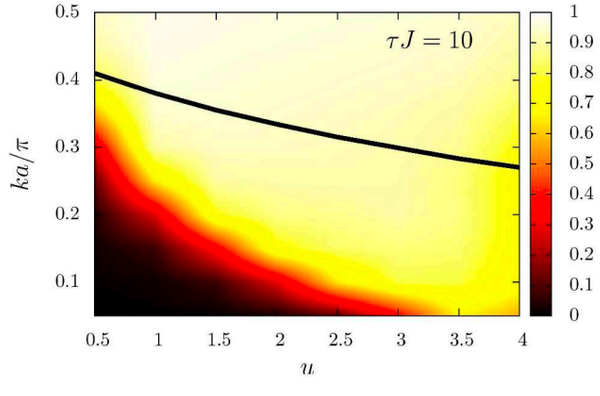
- **Part 1:** Tensors and “tensor-networks”

- **Part 2:** Bi-partite entanglement and bi-partite state compression

- **Part 3:** Matrix product states (MPS) and the time-evolving block decimation algorithm (TEBD) and application examples



- **Example** of MPS simulations

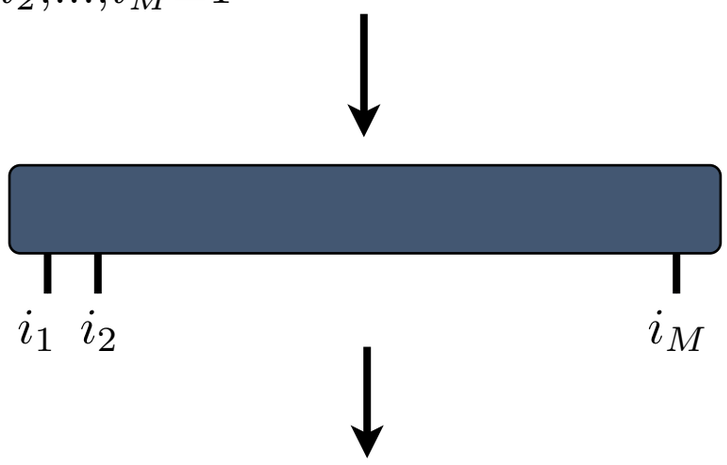


Matrix product states - The main idea

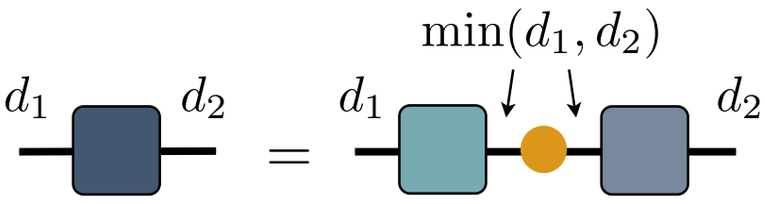
- Many-body quantum state (*M* local dimension *d*, e.g. qubits, bosons, ….):

$$= |\psi\rangle$$

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_M=1}^{d} c_{i_1,i_2,\ldots,i_M} |i_1, i_2, \cdots_M\rangle$$

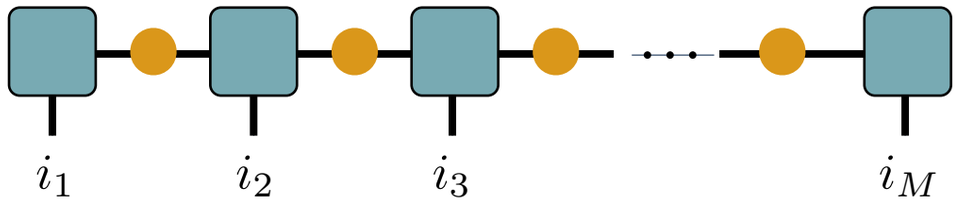


SVD:

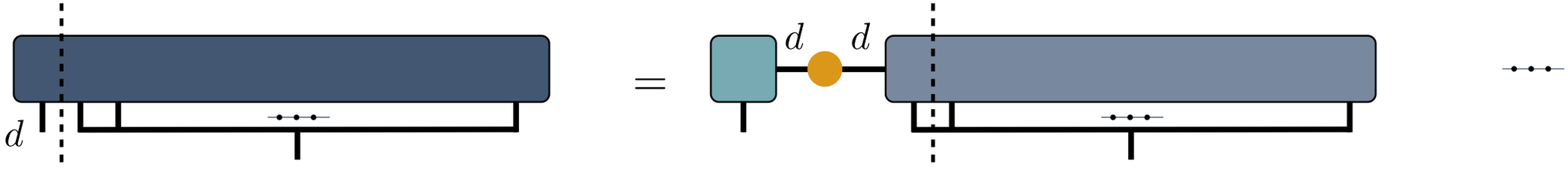


$$\mathbf{C} = \mathbf{USV}^\dagger$$

- **Matrix product state:**



- … can be done iteratively with SVDs

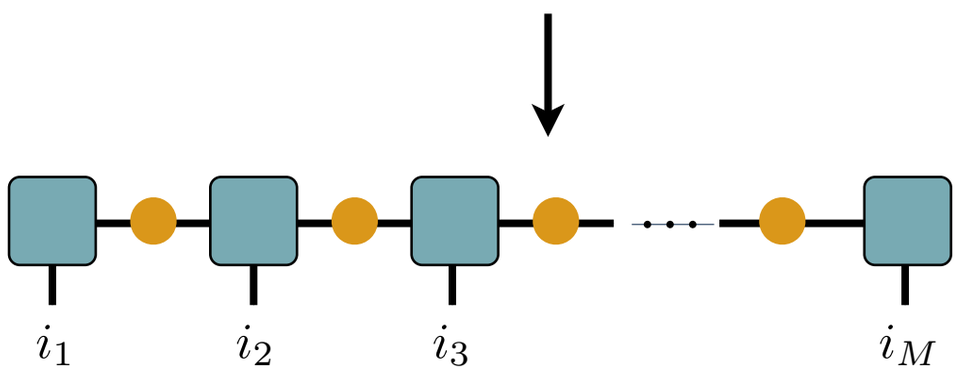


combine indices 2,…,M

iterate this

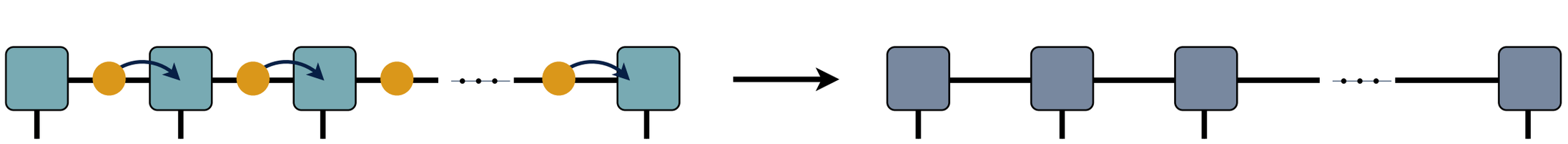
Matrix product states - The main idea

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_M=1}^{d} c_{i_1,i_2,\ldots,i_M} |i_1, i_2, \cdots M\rangle$$

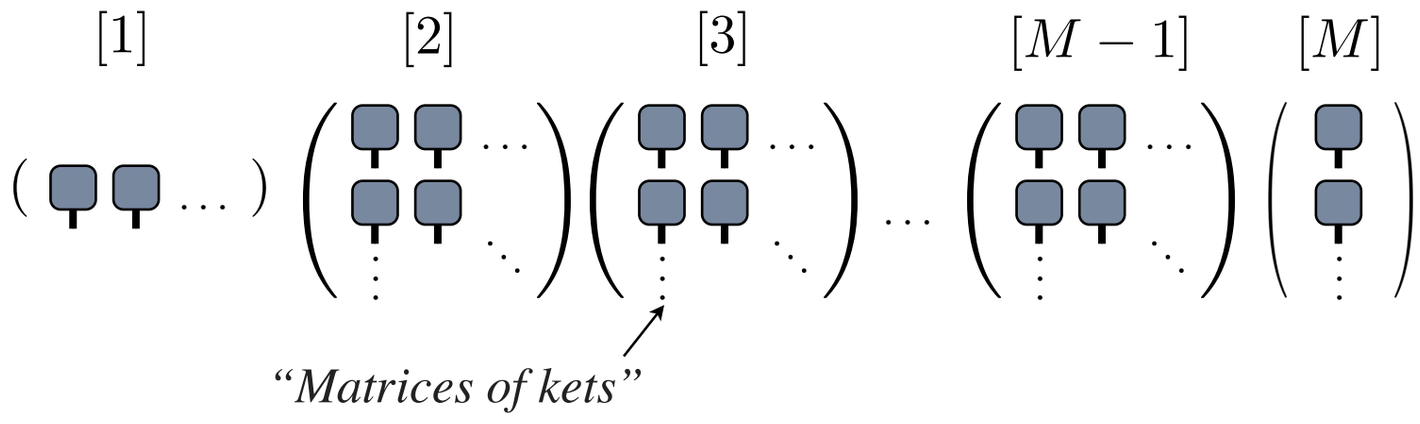


- **Matrix product state:**

- **Note 1:** Diagonal singular value matrices can be multiplied into other tensors



- **Note 2:** Another way to think about this decomposition:



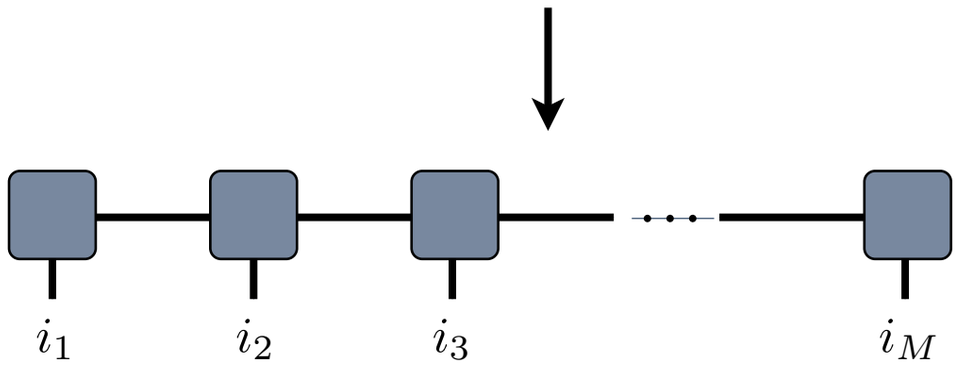
Matrix product

Hence the name: **Matrix product state**

Matrix product states - The main idea

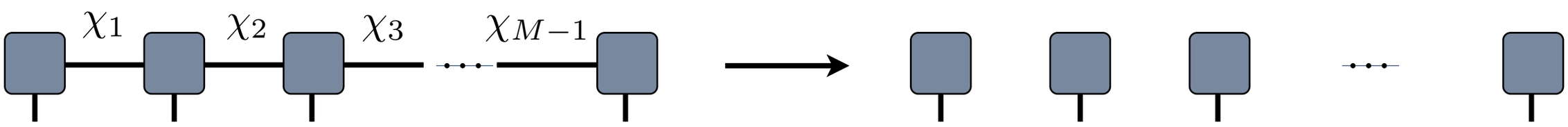
$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_M=1}^{d} c_{i_1,i_2,\ldots,i_M} |i_1, i_2, \cdots M\rangle$$

- **Matrix product state:**



- **Note 3:** A product state is a special case of a matrix product state:

If all horizontal dimensions are: $\chi_i = 1$

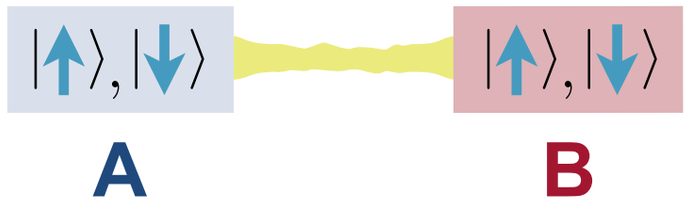


$c_{i_1,i_2,\ldots,i_M}$ $\qquad$ $c_{i_1} c_{i_2} \ldots c_{i_M}$

No entanglement!

Matrix product states - The main idea

- **Example:** Two spins



Entanglement: $|\psi\rangle \neq |\phi\rangle_A \, |\chi\rangle_B$

Product state

$$|\phi\rangle = |\uparrow\uparrow\rangle$$

$$\hat{\rho}_A = |\uparrow\rangle\langle\uparrow| \qquad S(\hat{\rho}_A) = 0$$

Entangled state

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle\right)$$

$$\hat{\rho}_A = \frac{\mathbb{1}}{2} = \frac{1}{2}|\uparrow\rangle\langle\uparrow| + \frac{1}{2}|\downarrow\rangle\langle\downarrow| \qquad S(\hat{\rho}_A) = 1$$

- Write states as "matrix product":

$$|\phi\rangle = \underset{1\times 1}{\left(|\uparrow\rangle\right)} \; \underset{1\times 1}{\left(|\uparrow\rangle\right)}$$

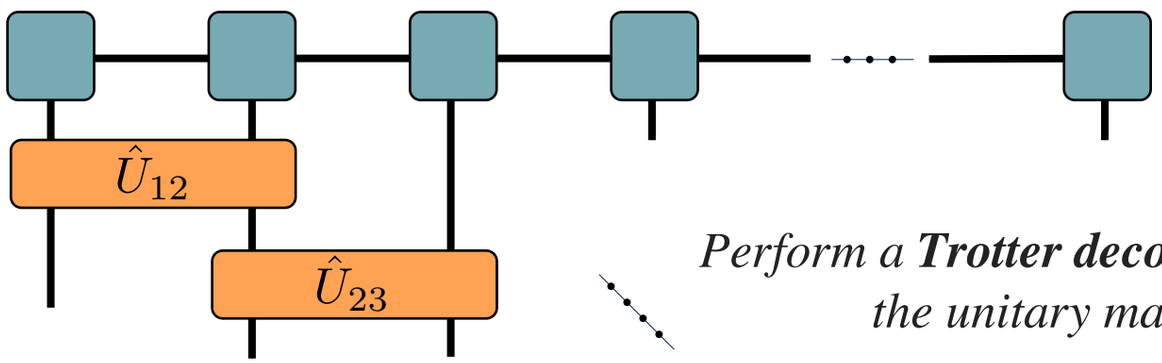
$$|\psi\rangle = \frac{1}{\sqrt{2}} \underset{1\times 2}{\left(|\uparrow\rangle \, |\downarrow\rangle\right)} \underset{2\times 1}{\begin{pmatrix}|\uparrow\rangle\\|\downarrow\rangle\end{pmatrix}}$$

- Key MPS idea: More entanglement $\Leftrightarrow$ Larger matrices needed

Sketch: TEBD algorithm

G. Vidal, Phys. Rev. Lett. 93, 040502 (2004)

- The conceptionally most simple way to simulate time evolution: The TEBD algorithm



$= |\psi(t)\rangle$

$$|\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}t\hat{H}} |\psi(t=0)\rangle$$

$$\hat{H} = \sum_{i,j} \hat{H}_{i,j} \quad \left[\hat{H}_{i,j}, \hat{H}_{k,l}\right] \neq 0$$

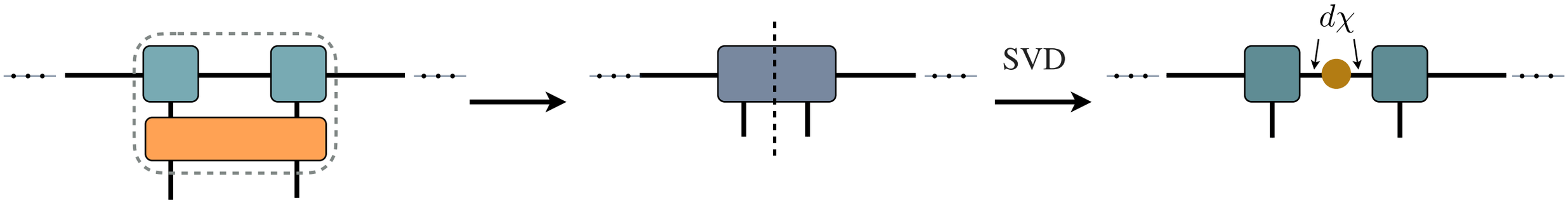
*Perform a **Trotter decomposition** of the unitary matrix:*

$$\mathrm{e}^{-\mathrm{i}\Delta t \sum_{i,j} \hat{H}_{i,j}} \approx \mathrm{e}^{-\mathrm{i}\Delta t \hat{H}_{1,2}} \mathrm{e}^{-\mathrm{i}\Delta t \hat{H}_{2,3}} \ldots$$

Order the terms to minimize the error from the non-cummutativity!

S. Lloyd, Science 273 (1996)

A. T. Sornborger, E. D. Stewart, Phys. Rev. A 60, 1956 (1999)



$\mathcal{O}(\chi^3 d^2)$

The new horizontal dimension can grow = *The gate builds up entanglement*

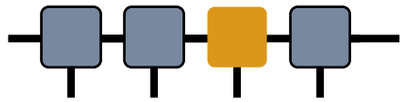
If the dynamics does not build up much entanglement between the *M-1* bipartitions, the error stays small and the simulation is exact!

Truncation needed! Error

$$\epsilon = \sum_{\alpha=\chi+1}^{d\chi} (s_{\alpha,\alpha})^2$$

Further reading … just scratching the surface in one lecture

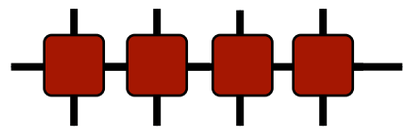
- Variational techniques
  (e.g. ground-state finding = **DMRG**)



e.g. local updates to minimize energy

"density matrix renormalization group" *S. White, 1992*

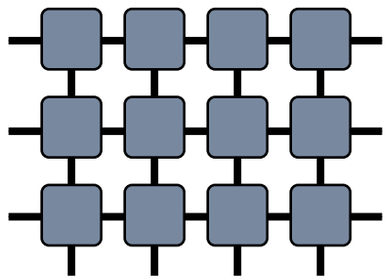
- Long range interactions: Matrix product operators (MPO) of **Hamiltonians** $\hat{H} =$



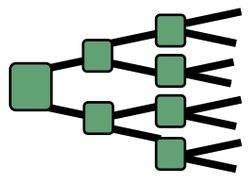
"projected entangled pair states"

- Elegant formulations for higher dimensional models (PEPS)

2D

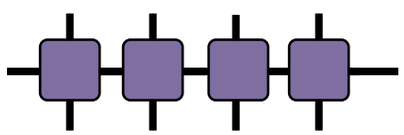


- Multi-scale entanglement renormalization ansatz (MERA):

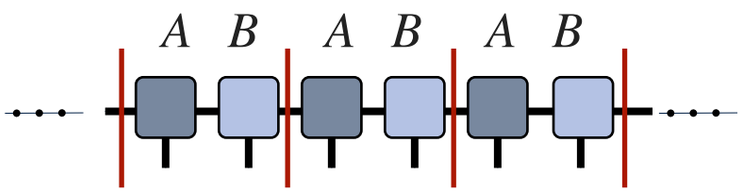


"Tensor network trees"

- Open systems: Density matrix as MPO $\hat{\rho} =$



- Infinite systems (iTEBD) … exploiting translational symmetries:



- **Further reading:** Review article with references:

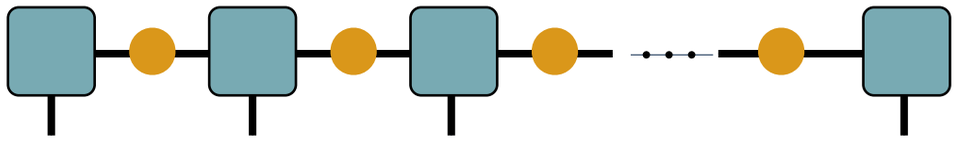
U. Schollwöck, Ann. Phys. 326, 96 (2011), arXiv:1008.3477

This time

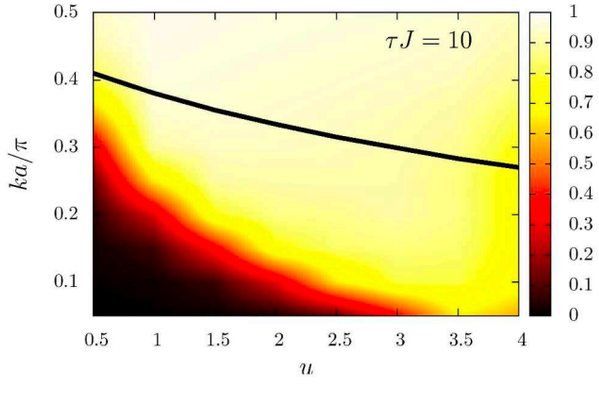
- **Part 1:** Tensors and “tensor-networks”

- **Part 2:** Bi-partite entanglement and bi-partite state compression

- **Part 3:** Matrix product states (MPS) and the time-evolving block decimation algorithm (TEBD) and application examples

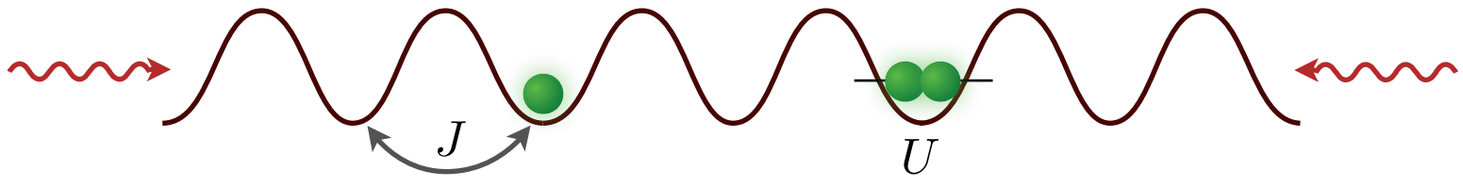


- **Examples:** MPS simulations



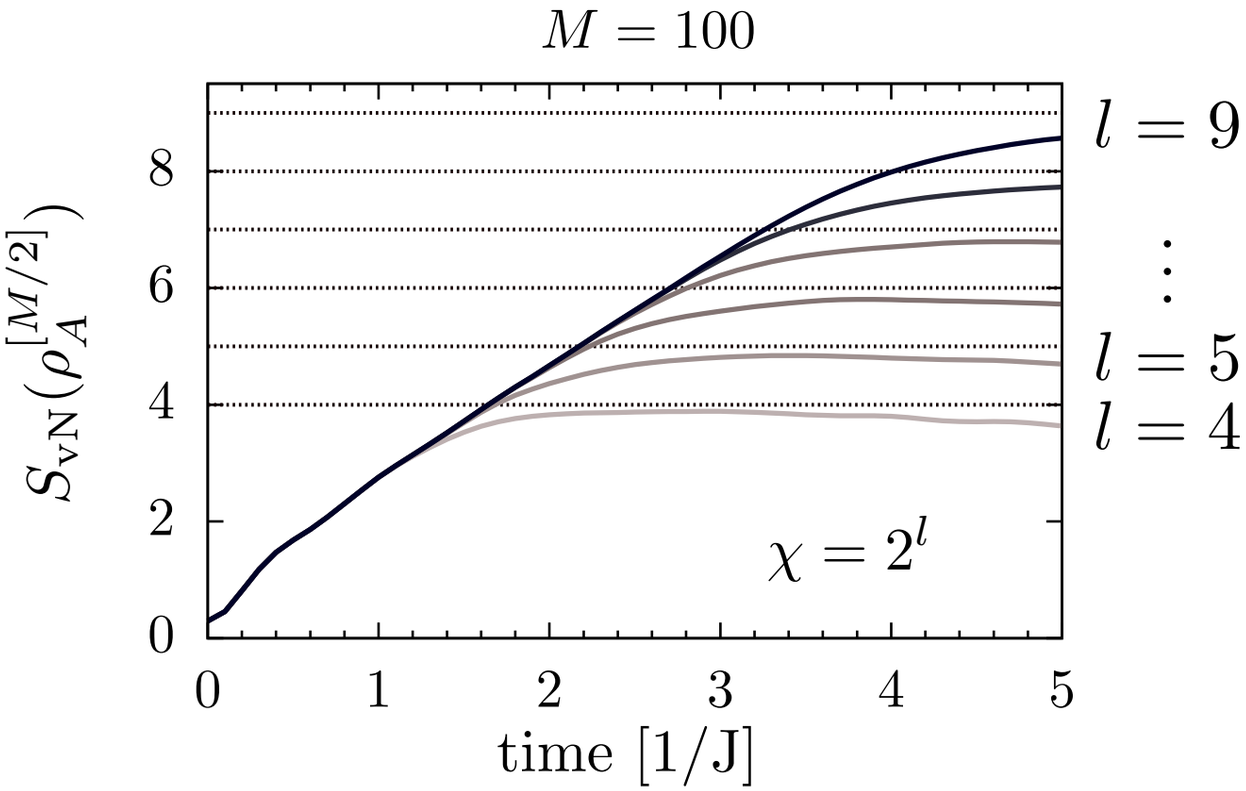
Example: Bose-Hubbard evolution

- **Model:** Bosonic atoms trapped in an optical lattice



$$\hat{H} = -J \sum_i (\hat{b}_i \hat{b}_{i+1}^\dagger + \hat{b}_i^\dagger \hat{b}_{i+1}) + \frac{U}{2} \sum_i \hat{n}_i(\hat{n}_i - 1) \qquad \hat{n}_i = \hat{b}_i^\dagger \hat{b}_i$$

- **Experiment:** Initial state  $U \gg J \;\; \rightarrow U \ll J$  *sudden quench!*



Linear growth

$$S_{\mathrm{vN}}(t) \propto t$$

$$S_{\mathrm{vN}}(t) \propto \log(\chi)$$

$$\chi \propto 2^t$$

Hard to simulate long times, but several tunneling times feasible

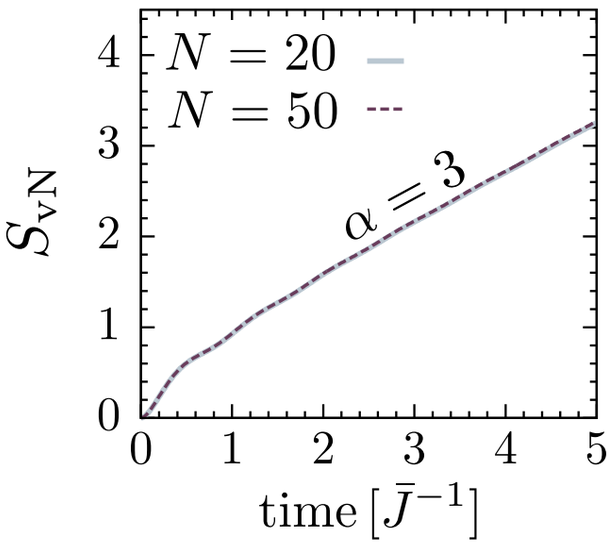
from A. J. Daley, H. Pichler, JS, P. Zoller, Phys. Rev. Lett. 109, 020505 (2012)

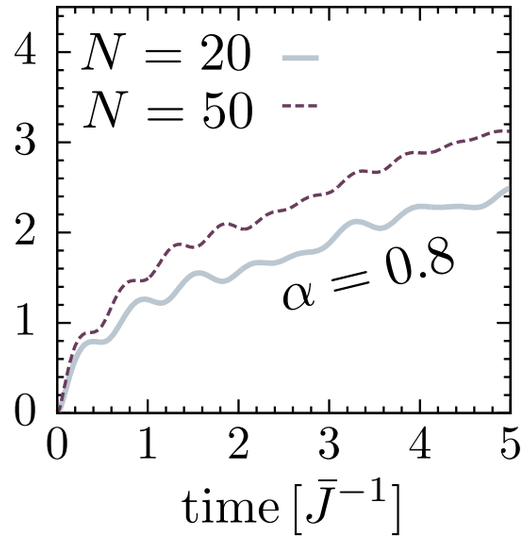
Example: Entanglement growth in long-range spin models

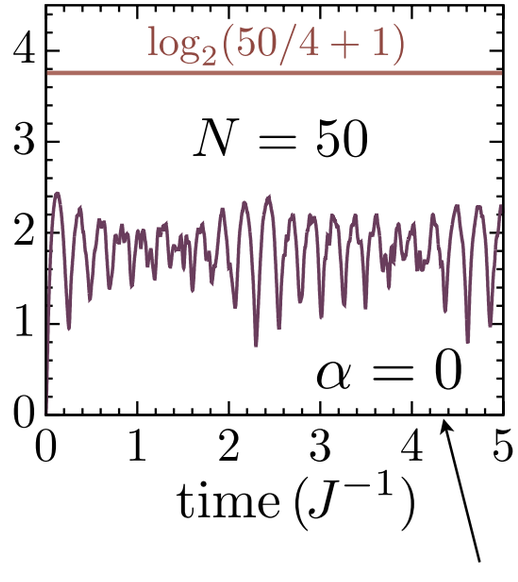
- Transverse Ising model, variable range (realized e.g. in trapped ions)

$$|\psi(t=0)\rangle = \bigotimes_i^N |\rightarrow\rangle \qquad \hat{H}_{\mathrm{TI}} = \sum_{j>i} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x\sum_i \hat{\sigma}_i^x \qquad J_{ij} = \frac{\bar{J}}{|i-j|^\alpha} \qquad h = \bar{J}$$

Short range	Long range	Infinite range







Permutation symmetric Hilbert sub-space (Dicke states)

JS, B. Lanyon, C. F. Roos, and A. J. Daley, Phys. Rev. X 3, 031015 (2013)

G. S. Bentsen, A. J. Daley, JS, "Entanglement dynamics in spin chains with structured long-range interactions", Springer (2022)

Reminder - GP simulations with Runge-Kutta

$$\frac{d}{dt}\psi(x,t) = -\mathrm{i}\left(\frac{\hat{p}^2}{2m} + V(x) + g|\psi(x,t)|^2\right)\psi(x,t)$$

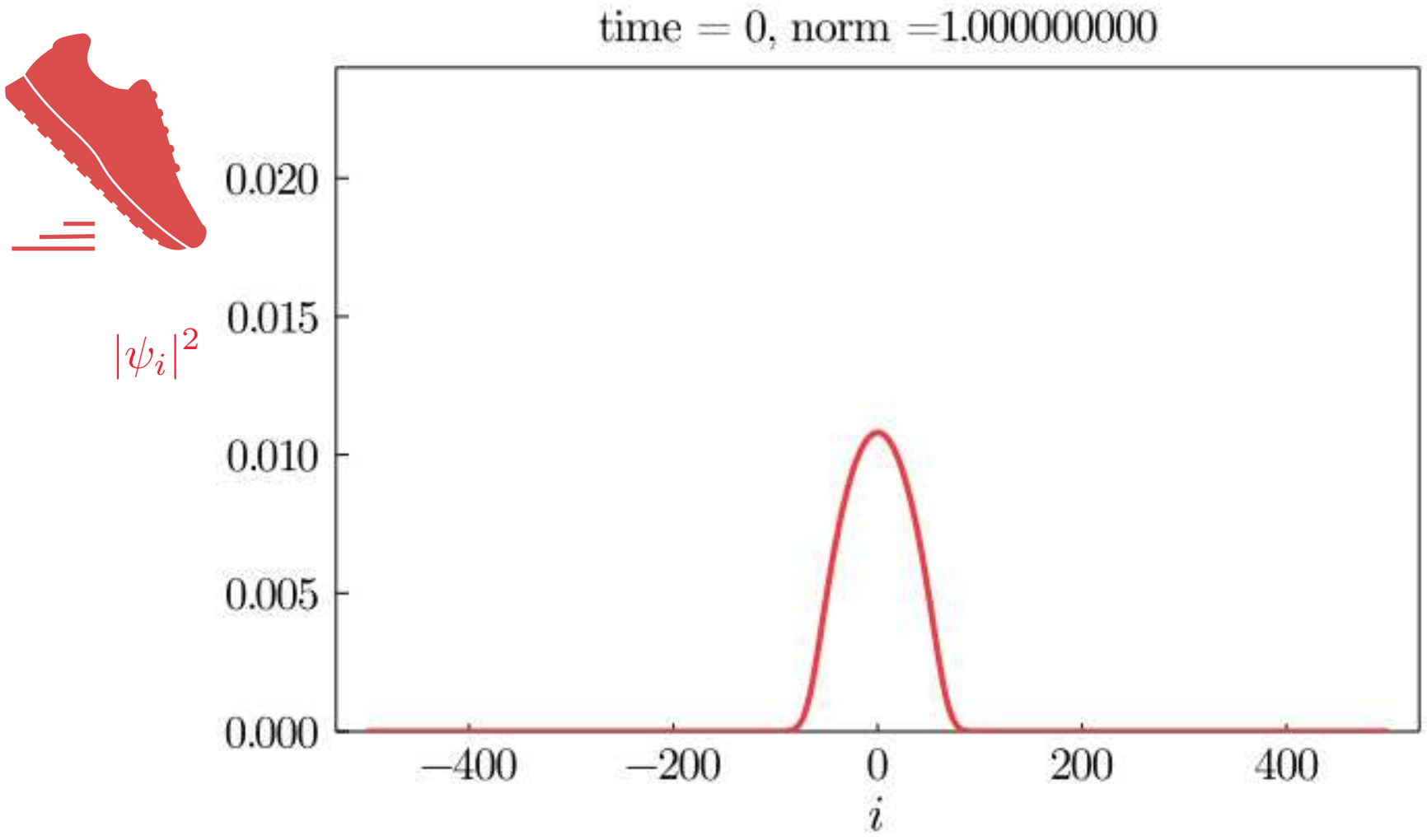
$$J = \frac{1}{2ma^2} \equiv 1$$

Solving it with RK4

$hJ = 0.02$

1001 grid points

periodic boundaries



$g = 5J$

$ka = 0.4\pi$

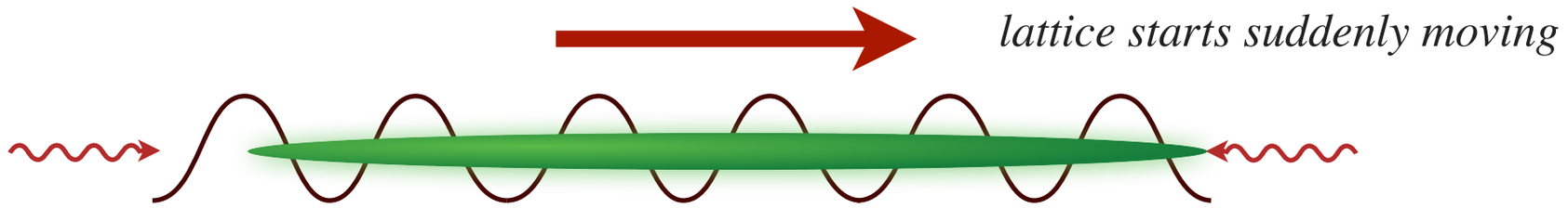
Our BEC get's destroyed!

This is known as dynamical instability!

Dynamical instability of a superfluid

- An experiment on stability of **superfluid currents** in the Bose-Hubbard model:

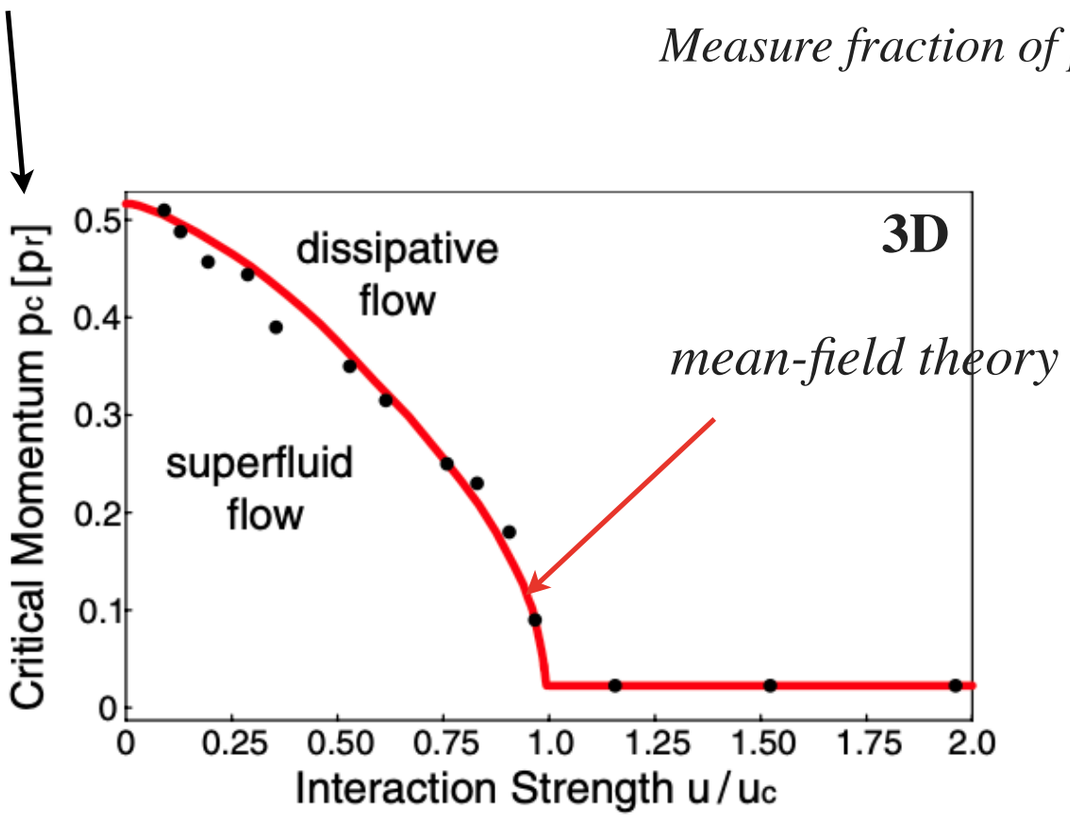
J. Mun, P. Medley, G. K. Campbell, L. G. Marcassa, D. E. Pritchard, and W. Ketterle, Phys. Rev. Lett. 99, 150604 (2007)



“momentum kick”

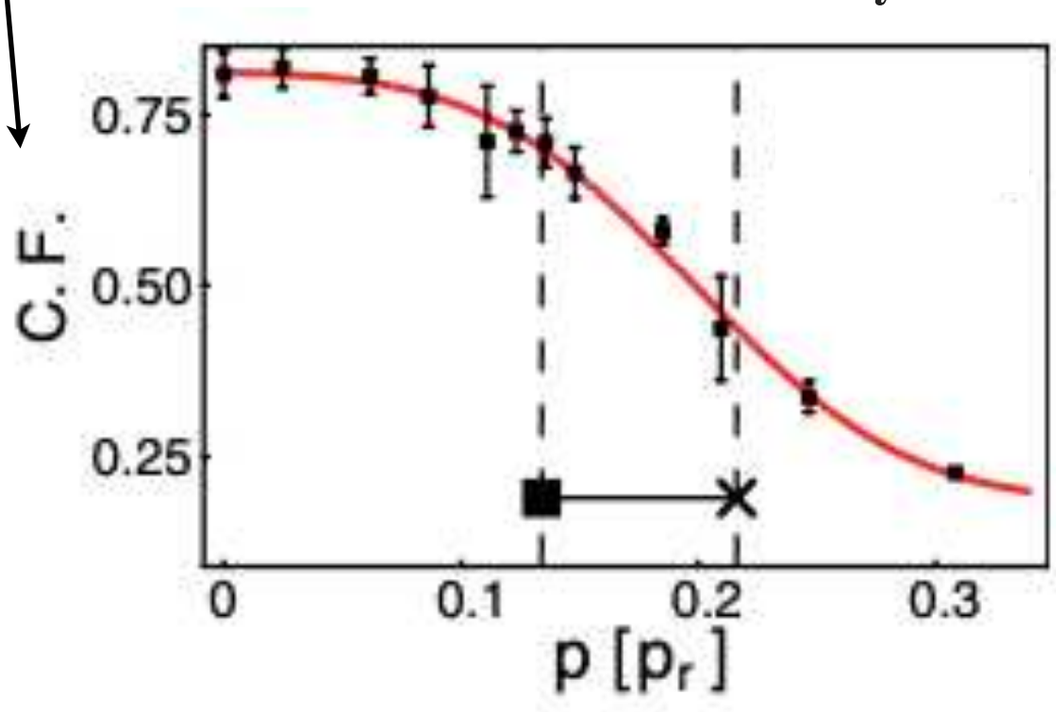
Measure fraction of particle in the condensate

…amount of kick



$u = U/J$

1D: mean-field failed badly

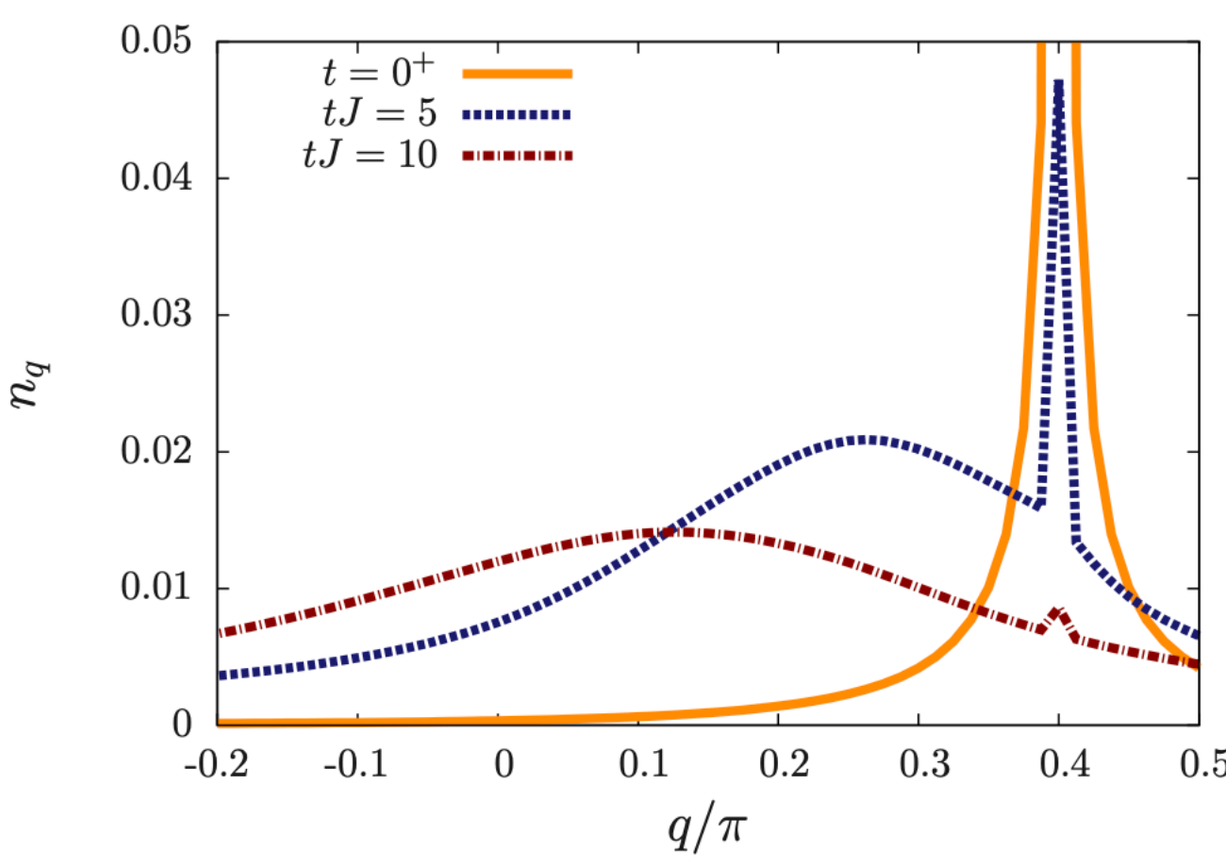


…no clear transition

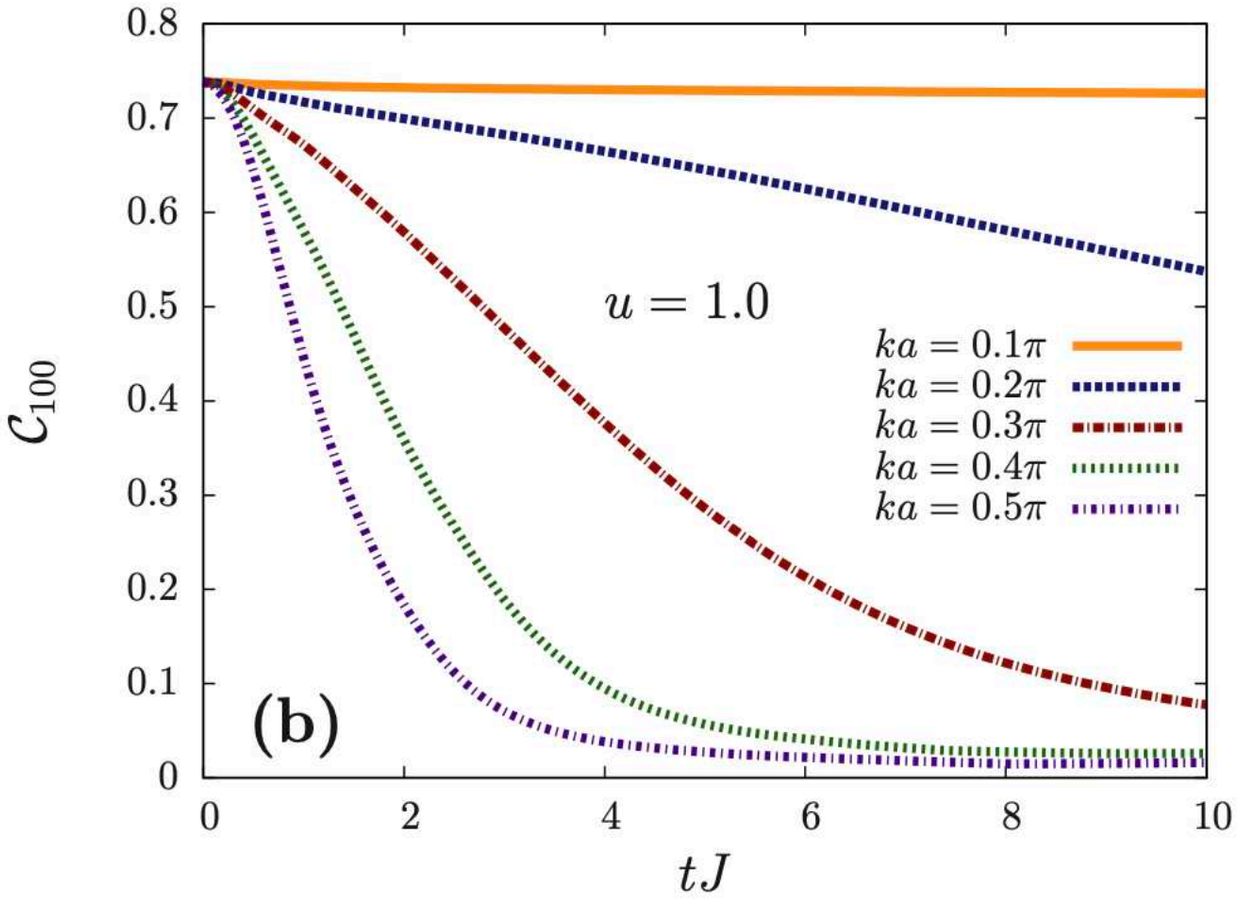
Dynamical instability of a superfluid

- **In 1D:** We can use MPS simulations

Momentum distributions



Condensate fraction

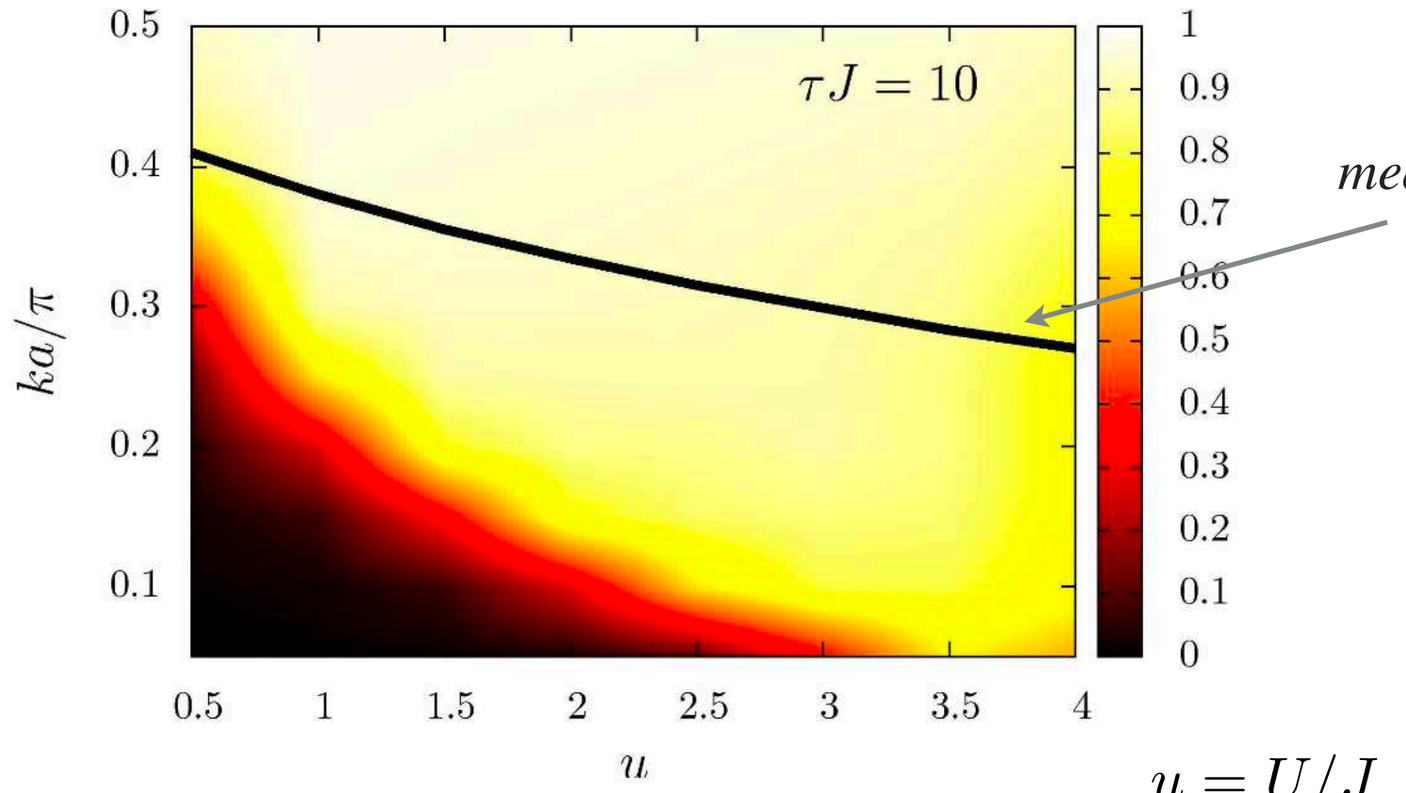


$u = U/J$

Dynamical instability of a superfluid

- We can use MPS to simulate the **loss in condensate fraction at a certain time**:

$$\Delta\mathcal{C}_R(\tau) \equiv |\mathcal{C}_R(t=\tau) - \mathcal{C}_R(t=0)|/\mathcal{C}_R(t=0)$$



mean-field does not work!

$u = U/J$

New J. Phys. 12, 025014 (2010)

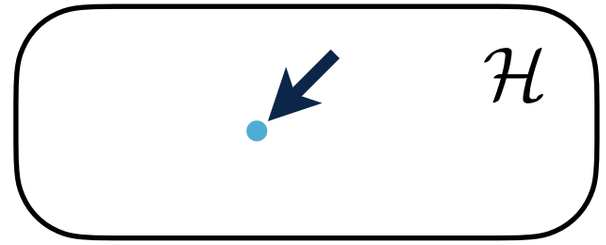
Take-away messages

A tour through numerical methods for simulating large-scale quantum physics (classically)

We discussed numbers and linear algebra on classical computers.

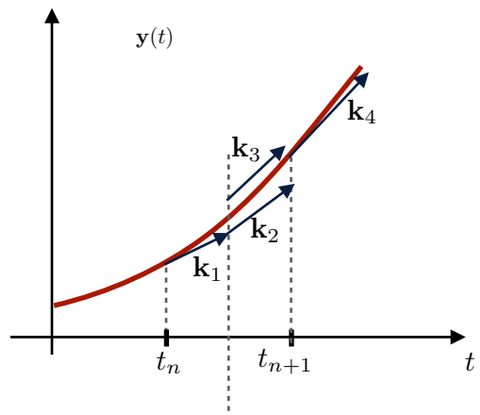
We discussed several numerical methods for the simulation of quantum dynamics:
Full quantum state dynamics is linear:
- Exact diagonalization solves the Schrödinger equation
- Physical Hamiltonians are sparse, **Krylov space methods** make use of that

In the first two lectures, we kept the quantum state-representation **exact!**
Then going to systems with more than 30 qubits/spins is hard.

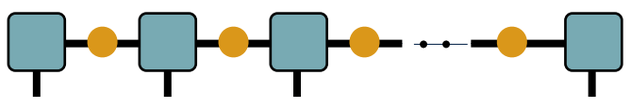


A **mean-field approximation** can drastically **reduce the state-space** but makes the problem **non-linear** (and it's a strong approximation).

Then, one needs general ODE solvers. We introduced **Runge-Kutta methods,** which are generally a "swiss army knife tool", especially the 4th order method



Mean-field relies on non-entangled product states. It's an the limit of a **matrix product state** with bond dimension zero.



Matrix product states: a systematic way to compress information! The crucial quantity that makes classical simulations hard is **not system size,** but entanglement entropy!