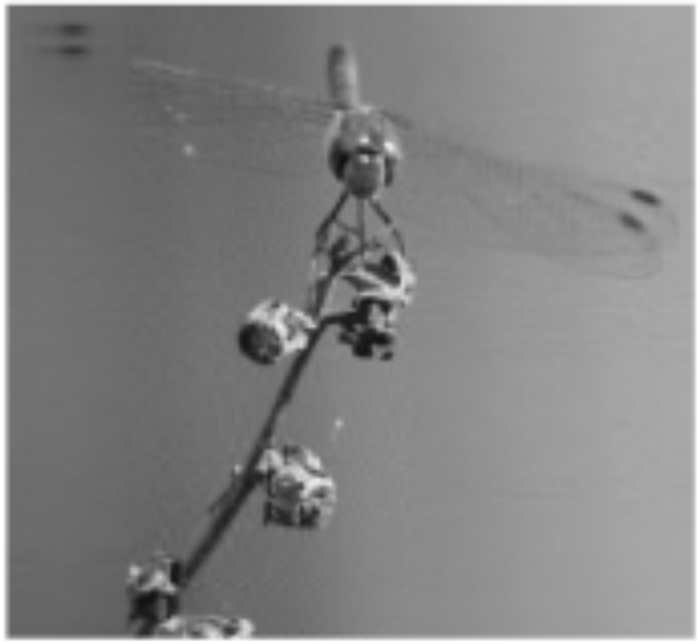
Simulate Quantum Computers with Matrix Product States

Department of Physics and Astronomy
Università degli Studi di Padova

Marco Ballarin
marco.ballarin.6@phd.unipd.it

Goals

- Present the Matrix Product State simulation method for quantum computers;
- Understand how the simulator works;
- Understand in which cases it makes sense to use the simulator;
- Implement some parts of it during the hands on!

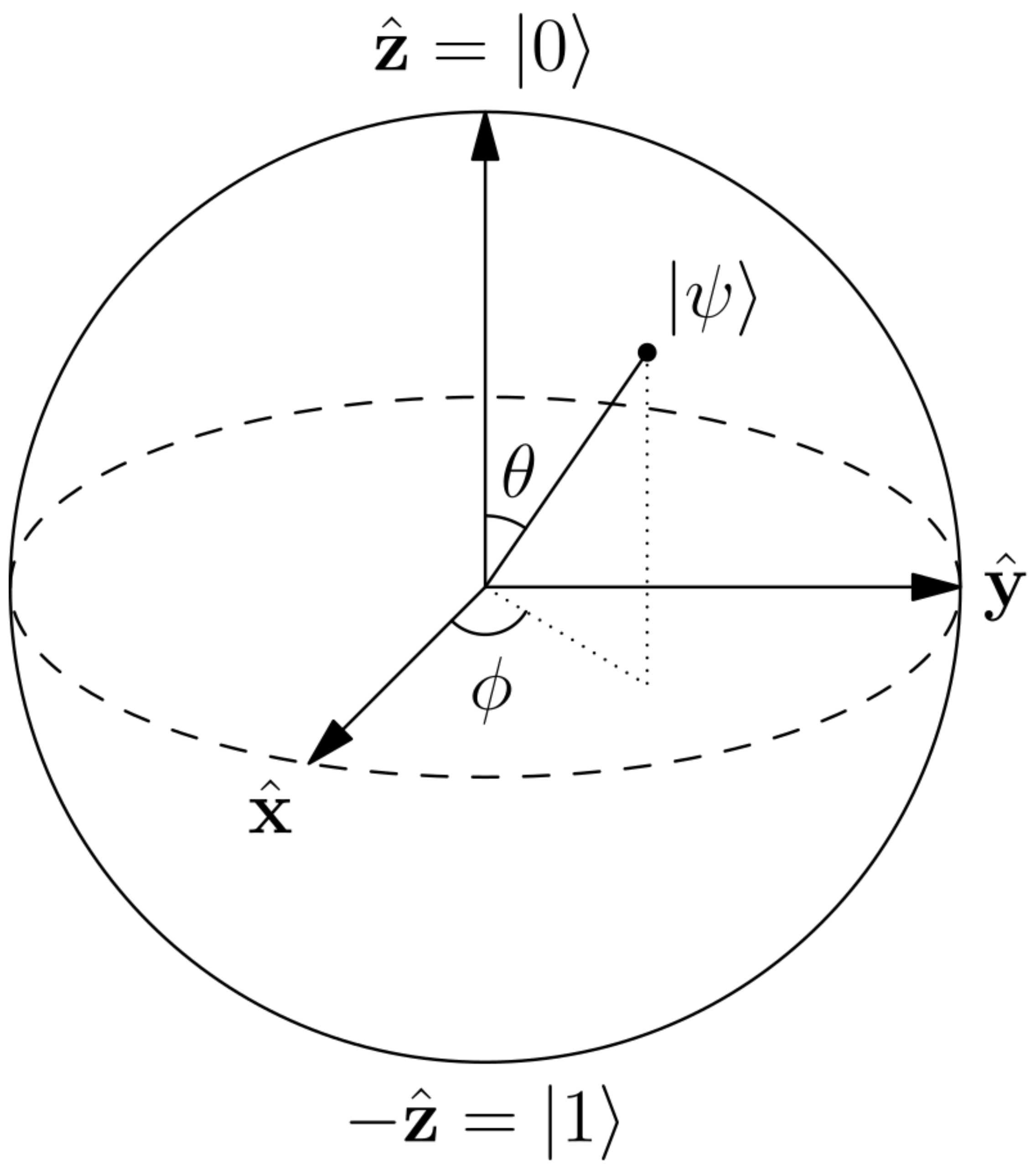

Introduction

Classical bit $b \in \{0,1\}$

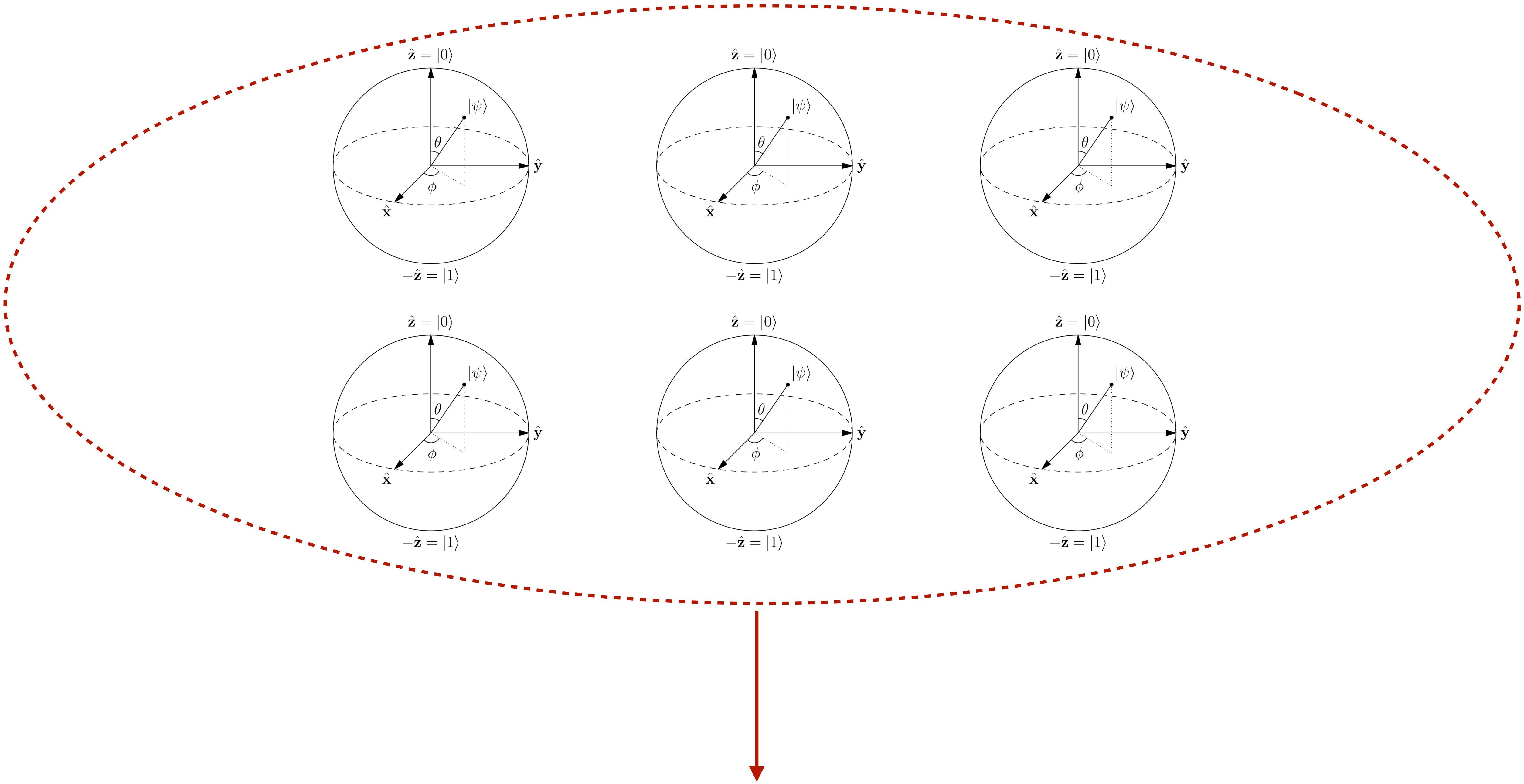
$\downarrow$

quantum qubit $|\psi\rangle \in \mathscr{H}, \dim(\mathscr{H}) = 2$

$$|\psi\rangle = \cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle$$



Problem: $\dim(\mathscr{H})$



$$\mathbf{dim}(\mathscr{H}) = 2^6 = 64$$

Exponential scaling

In general, we have:

$$\mathbf{dim}(\mathscr{H}) = 2^n$$

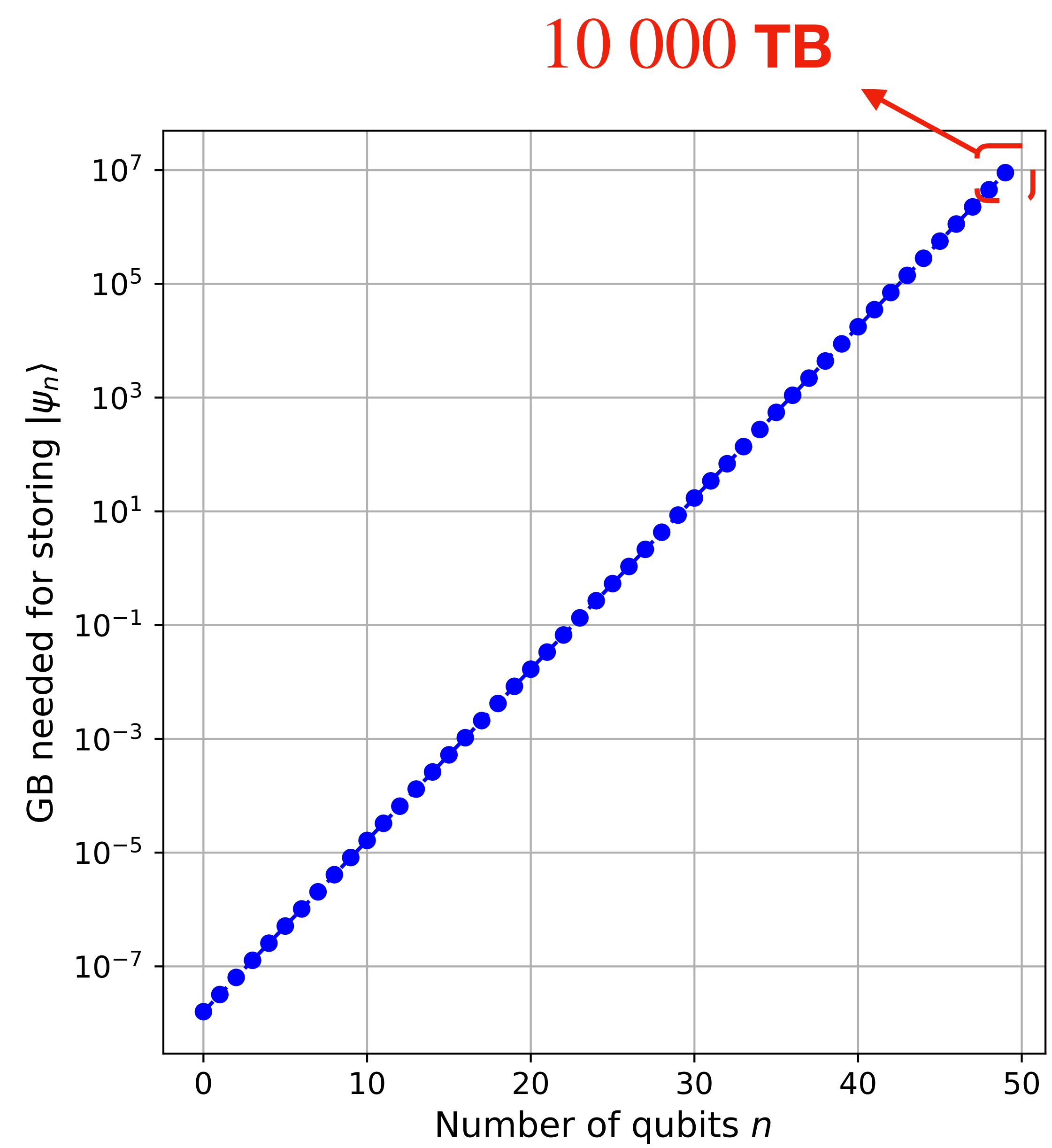
Complex numbers

Double precision

$$\mathbf{GB}\left(|\psi\rangle\right) = \frac{2 \cdot 64}{8}\mathbf{dim}(\mathscr{H}) \cdot 10^{-9}$$

bit → byte

byte → Gigabytes



Compress information

CINECA

String compression (simple example):

AAABBCDDDDDDDD **Length: 14**

Compress

A3B2C1D8 **Length: 8**

RANDOM

Compress

How much we can compress a classical string?

Shannon entropy $\boxed{H(X)} = -\sum_{i}^{n} p(x_i) \log p(x_i)$ x_i digit

The **bigger** is the entropy the **smaller** is the compression

Hands on: compression

Can we compress quantum resources as we did for the strings? And what resource?

https://www.menti.com/whfzqpxizx

Try it yourself!

- Open the jupyter notebook “Compression.ipynb”
- Using the given function try to understand which states are easier to compress
- 5 minutes of time!

Entanglement!

To efficiently compress a quantum state we need to compress its entanglement!
But how do we quantify the entanglement?

$$\rho = |\psi\rangle\langle\psi|$$

Pure state
Defines everything about our system

Trace away a part of the system
$\Rightarrow$ sum over all possible values of that
Part of the system

Equivalent to marginalize the probability

$$p(x) = \int dy p(x, y)$$

ρ_A

We define the entanglement between the bipartitions ρ_A, ρ_B as the Von Neumann Entropy:

$$S_V(\rho) = -\mathbf{Tr}\left(\rho_A \log \rho_A\right) = -\sum_{i=1}^{\chi} \lambda_i \log \lambda_i$$

λ_i eigenvalues of ρ_A
$\sum_i \lambda_i = 1$

Entanglement!

We define the entanglement between the bipartitions ρ_A, ρ_B as the Von Neumann Entropy:

Number non-zero eigenvalues

First indicator of the entanglement

$$S_V(\rho) = -\mathbf{Tr}\left(\rho_A \log \rho_A\right) = -\mathbf{Tr}\left(\rho_B \log \rho_B\right) = -\sum_{i=1}^{\chi} \lambda_i \log \lambda_i$$

How much entanglement had the states we studied in the hands on?

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \dots \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$$

Product state

Not entangled

$\chi = 1$

$$S_V(\rho) = 0$$

$$\frac{1}{\sqrt{2}}(|0\dots0\rangle + |1\dots1\rangle)$$

Slightly entangled

$\chi = 2$

$$S_V(\rho) = \log 2$$

$$|random\rangle$$

Highly entangled

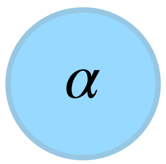
$\chi \simeq 2^n$

$$S_V(\rho) \simeq \log \chi$$

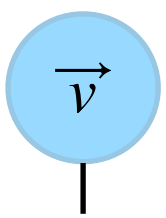
$S_V(\rho)$ is maximised $\iff \lambda_i = \lambda_j \forall i, j \Rightarrow \lambda_i = \dfrac{1}{\chi}$: $\longrightarrow$ $\max(S_V) = \log \chi$

From the intuition to the application

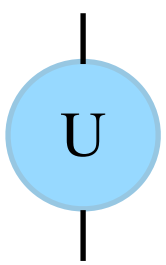
Tensors



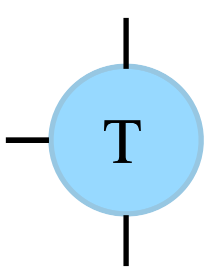
= order-0 tensor = **scalar**



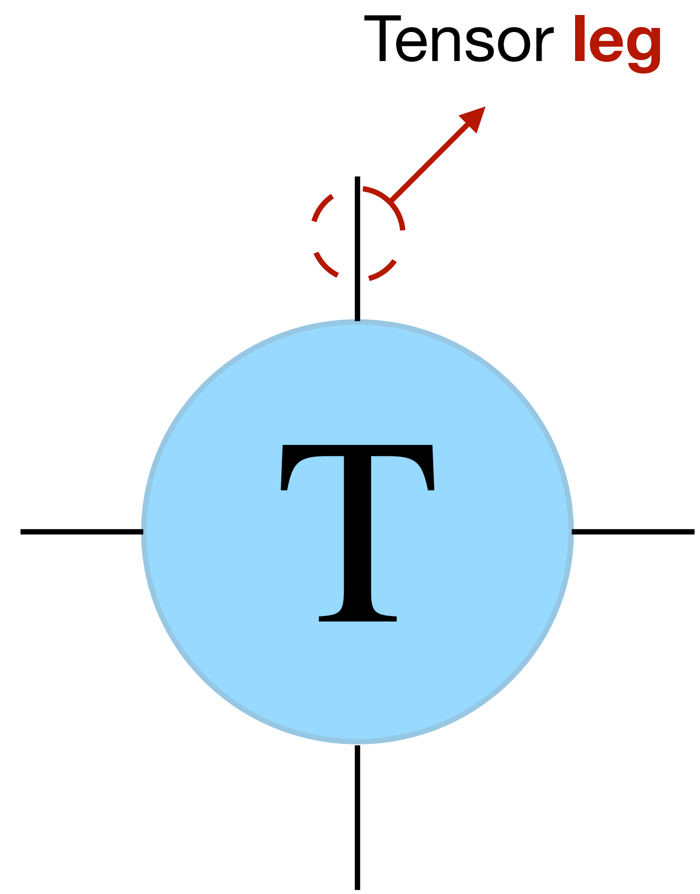
= order-1 tensor = **vector**



= order-2 tensor = **matrix**

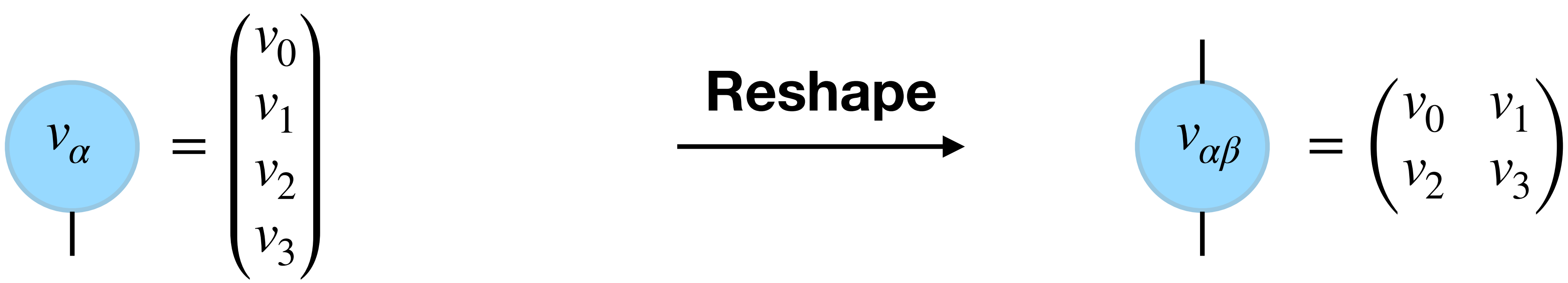


= order-3 tensor = **tensor**

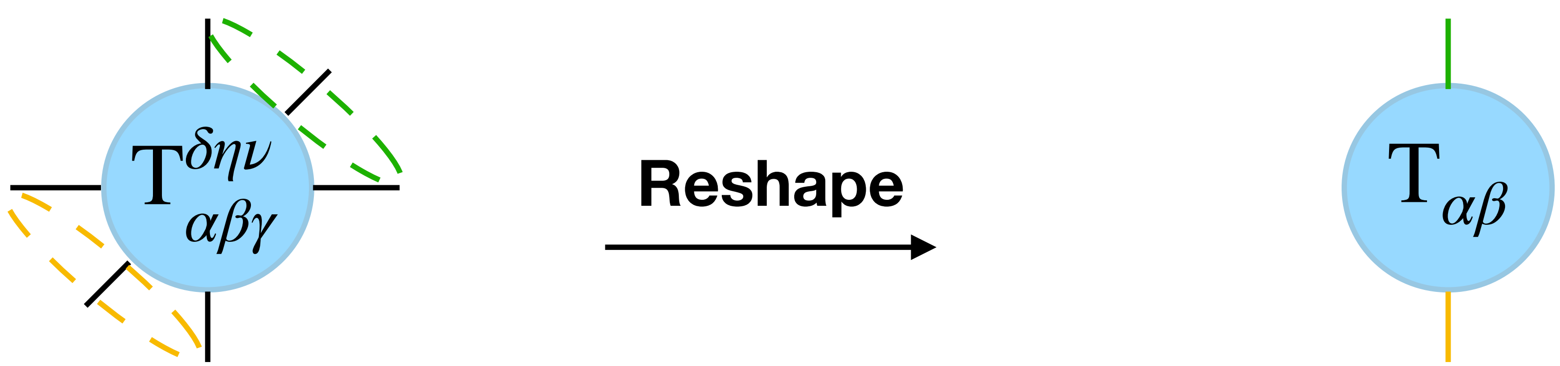


Tensor Manipulation

We can manipulate **Tensors** and **reshape** their indexes (legs) as we prefer:



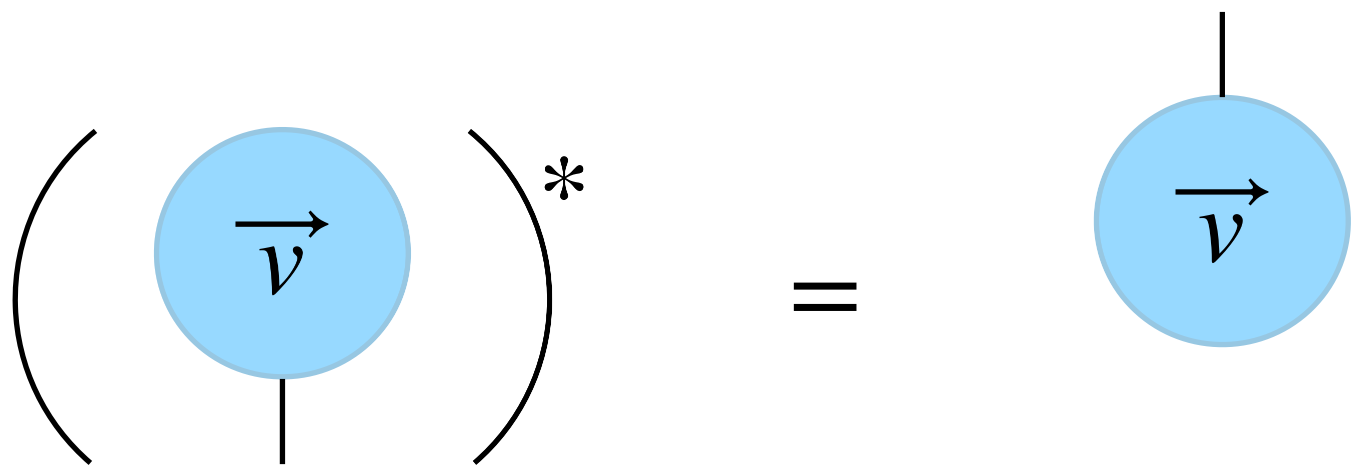
This means that a tensor of **any order** can be mapped to a **matrix**. So, we can use linear algebra to work with tensors.



Tensor operations

We can perform operations on tensors, and we have to decide a notation for them, in particular using the graphical notation introduced previously.

We start by introducing the **complex conjugate** of a order-1 tensor:



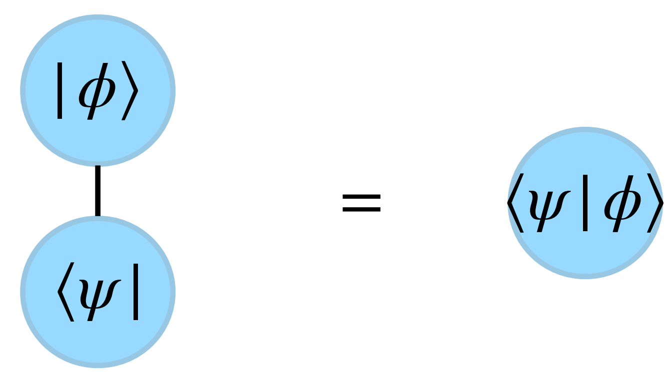
Tensor operations

We can perform operations on tensors, and we have to decide a notation for them, in particular using the graphical notation introduced previously.

Then we introduce the **contraction** between two tensor along their **legs.**

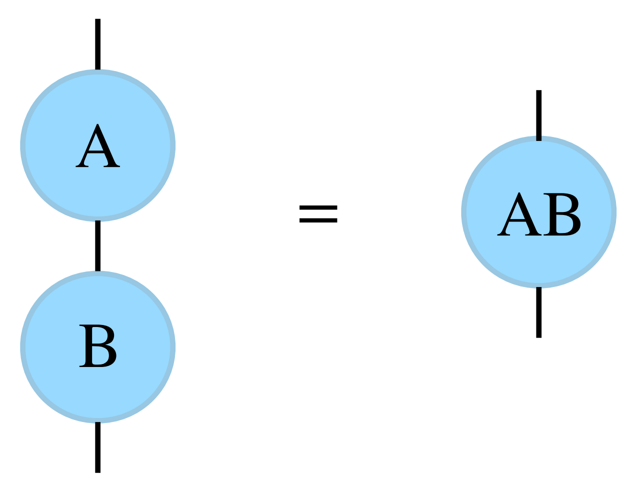
- We start from two order-1 tensor, and it is equivalent to the scalar product between two vectors:

$$\langle \psi | \phi \rangle = \sum_i \psi_i^* \phi_i =$$



- Then, the contraction between two order-2 tensor is simply the matrix-matrix multiplication:

$$(\mathrm{AB})_{ik} = \sum_j a_{ij} b_{jk} =$$

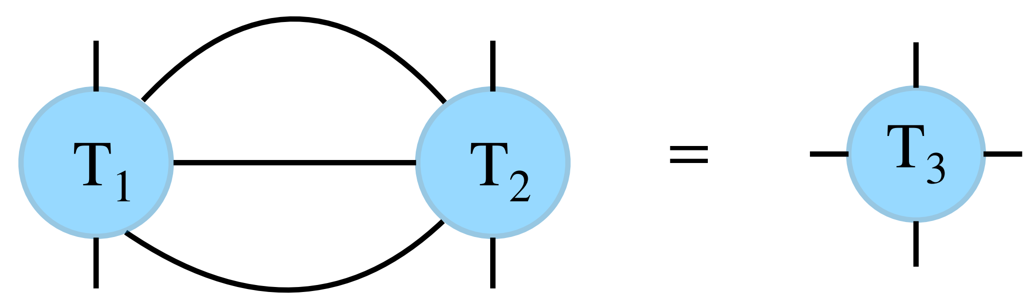


Contraction

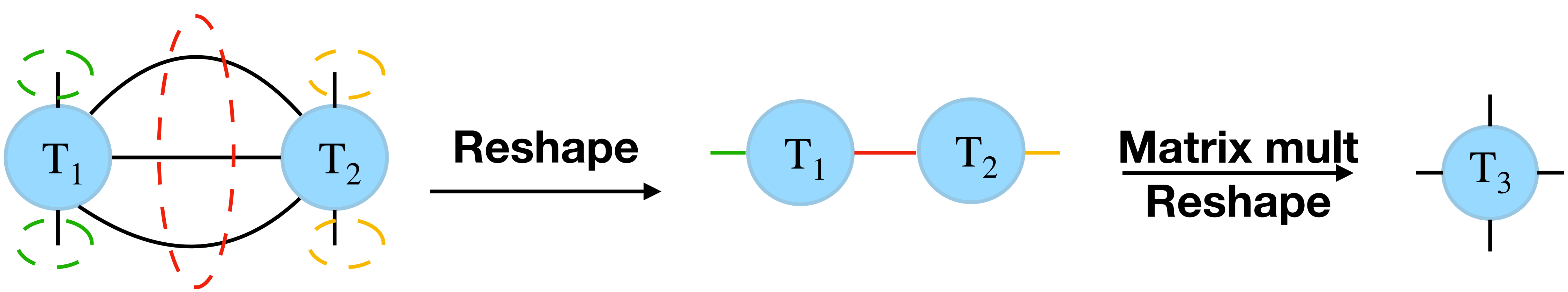
We can perform operations on tensors, and we have to decide a notation for them, in particular using the graphical notation introduced previously.

- In general, we can contract any leg of an order-n tensor:

$$T_3 = \sum_{\alpha\beta\gamma} \mathrm{T}_{1,\alpha\beta\gamma\delta\eta} \mathrm{T}_{2,\alpha\beta\gamma\mu\nu} =$$



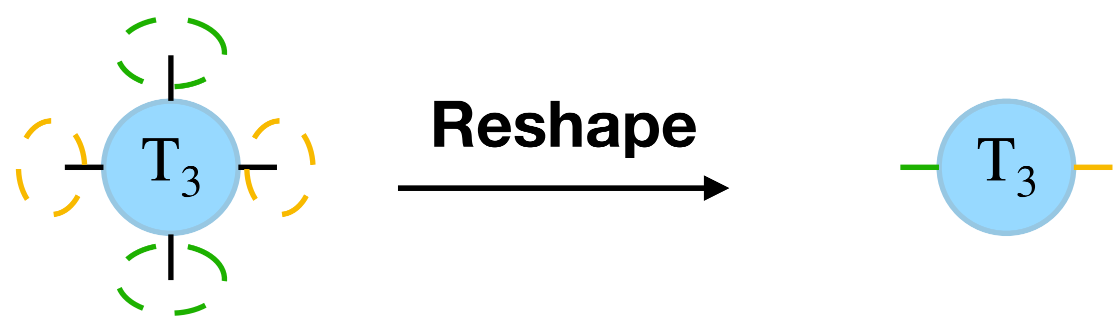
- What will be done in practice by the simulator, however, will be a little different:



SVD

We can perform operations on tensors, and we have to decide a notation for them, in particular using the graphical notation introduced previously.

- Finally, we present a way to **split** tensors. This means that we can pass from a single tensor to two tensors. First, we reshape it such that we have a matrix, dividing the indices that we want to divide

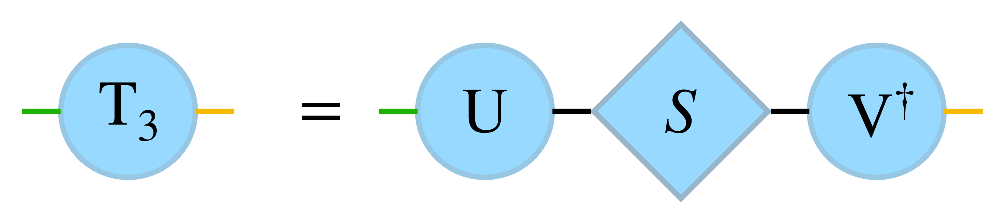


- Then, we use the **Singular Value Decomposition** (SVD) technique to separate the tensor. We obtain three matrices:

Unitary

$$T_3 = USV^\dagger$$

Diagonal

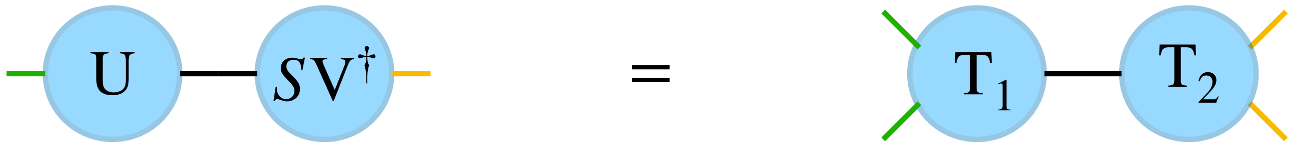


SVD

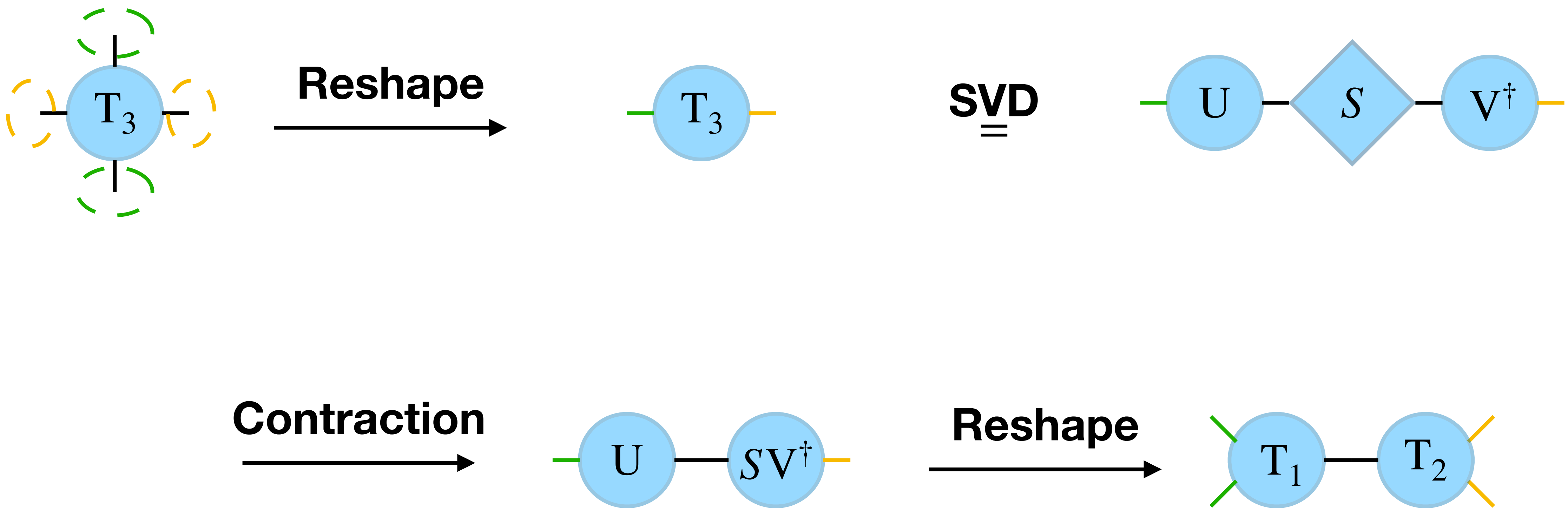
We can perform operations on tensors, and we have to decide a notation for them, in particular using the graphical notation introduced previously.

- Then, we contract the diagonal matrix S with $V^\dagger$. We thus end up with two matrices:

- Finally, we reshape the tensors to have the original legs (2 green, 2 yellow)



SVD: RECAP



Try it yourself!

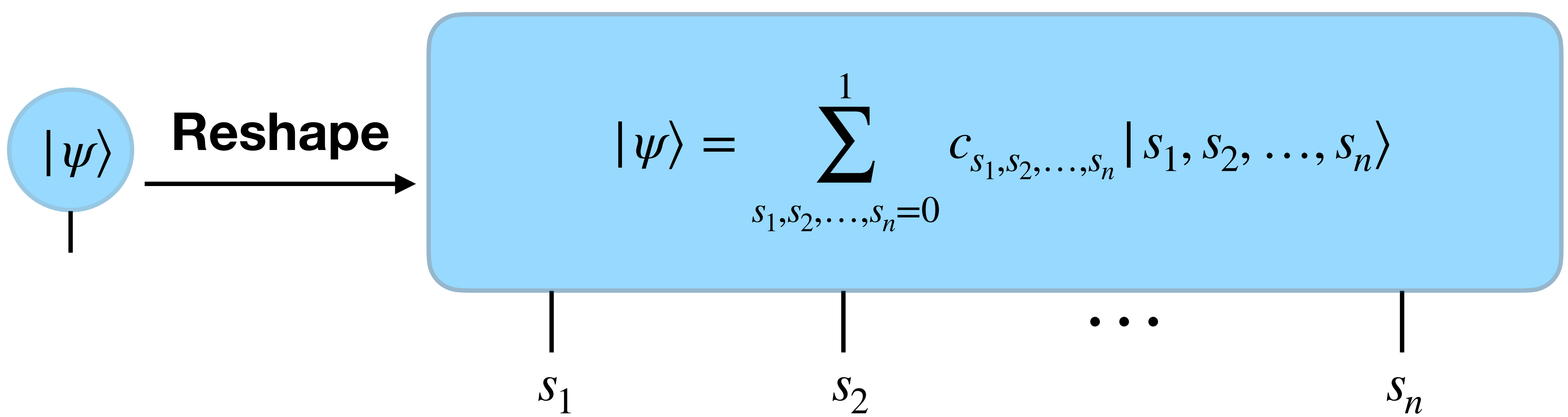
- Open the jupyter notebook “TensorSVD.ipynb”
- Using the given function try to write down an SVD which performs the operation above given a 4-legs tensor
- 10 minutes of time!

Break

Quantum State

We can finally come back to the quantum computation framework. We will so consider an n-qubits state $|\psi\rangle \in \mathcal{H}$, with $\dim(\mathcal{H}) = 2^n$.

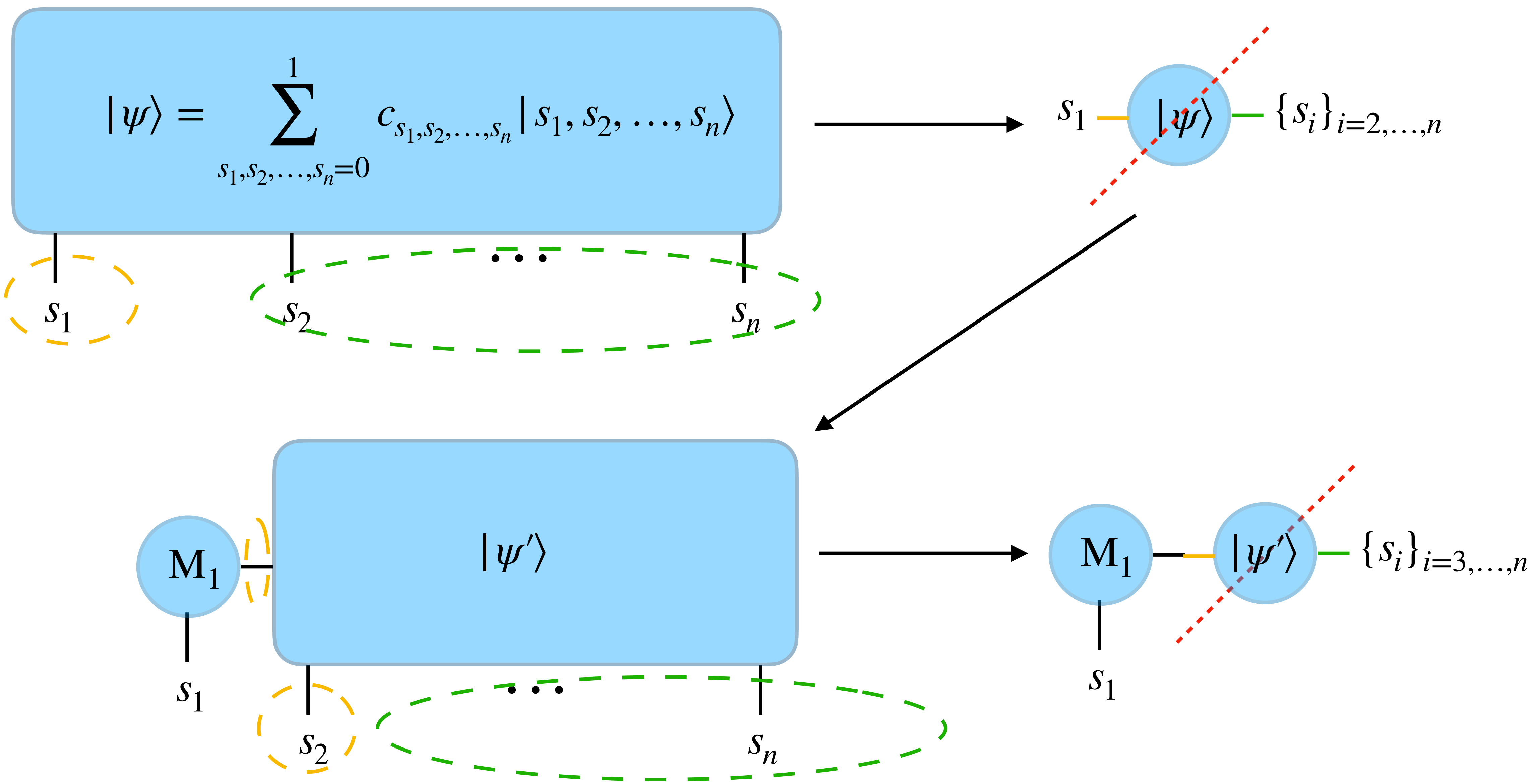
- A quantum state can be represented as a vector. We can reshape that vector as an order-n tensor, where each leg has dimension 2.



- We can then apply iteratively the **splitting** of the tensor, as seen previously.

Quantum State

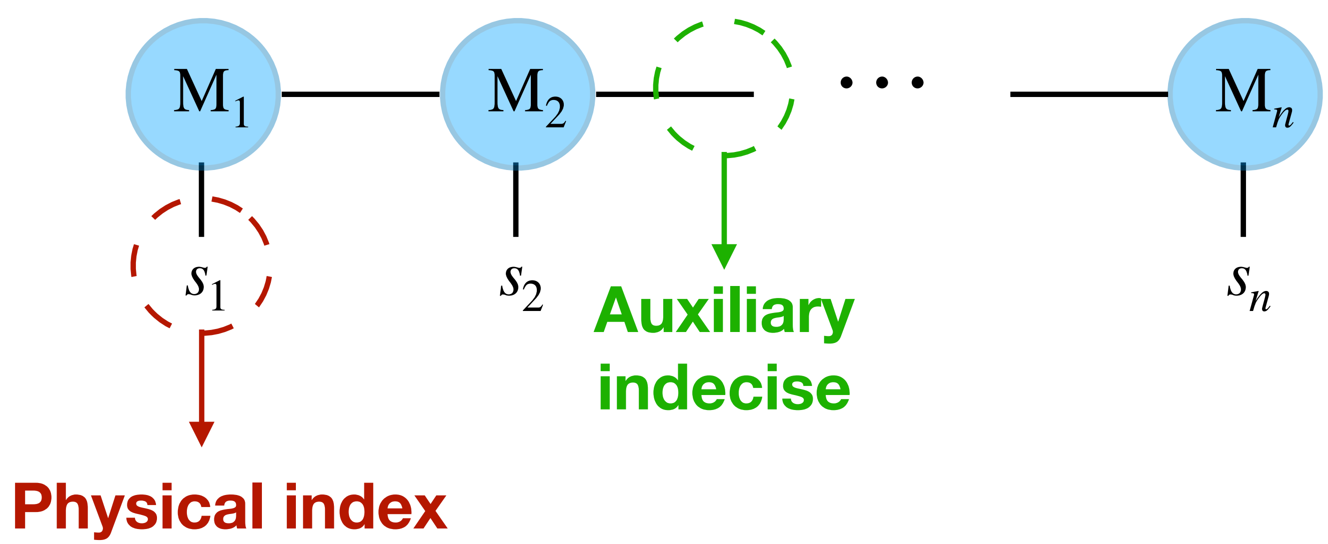
We can finally come back to the quantum computation framework. We will so consider an n-qubits state $|\psi\rangle \in \mathscr{H}$, with $\dim(\mathscr{H}) = 2^n$.



Quantum State

We can finally come back to the quantum computation framework. We will so consider an n-qubits state $|\psi\rangle \in \mathcal{H}$, with $\dim(\mathcal{H}) = 2^n$.

- We end up with a network of $n-2$ order-3 tensor and 2 order-2 tensor at the boundaries:



- Are we getting a memory advantage by using this decomposition of the state?

Hands on!

Try it yourself!

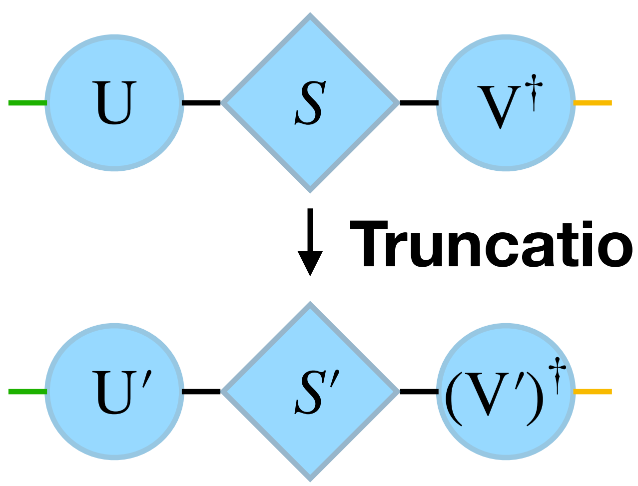
- Open the jupyter notebook “qstate.ipynb”
- Copy your SVD implementation or use the available one
- Try to transform your quantum state in tensors, and look at the amount of memory you need to store them
- 5 minutes of time!

https://www.menti.com/hh2gv2waoo

Truncation

We can finally come back to the quantum computation framework. We will so consider an n-qubits state $|\psi\rangle \in \mathscr{H}$, with $\dim(\mathscr{H}) = 2^n$.

- We call χ_{max} **bond dimension** of the system, and denote with s_1 the greatest eigenvalue of S. Then:



$$S' = \begin{cases} s_i & \textbf{if } \frac{s_i}{s_1} \leq \epsilon \textbf{ and } i \leq \chi_{max} \\ 0 & \textbf{otherwise} \end{cases}$$

We then truncate the 0 term

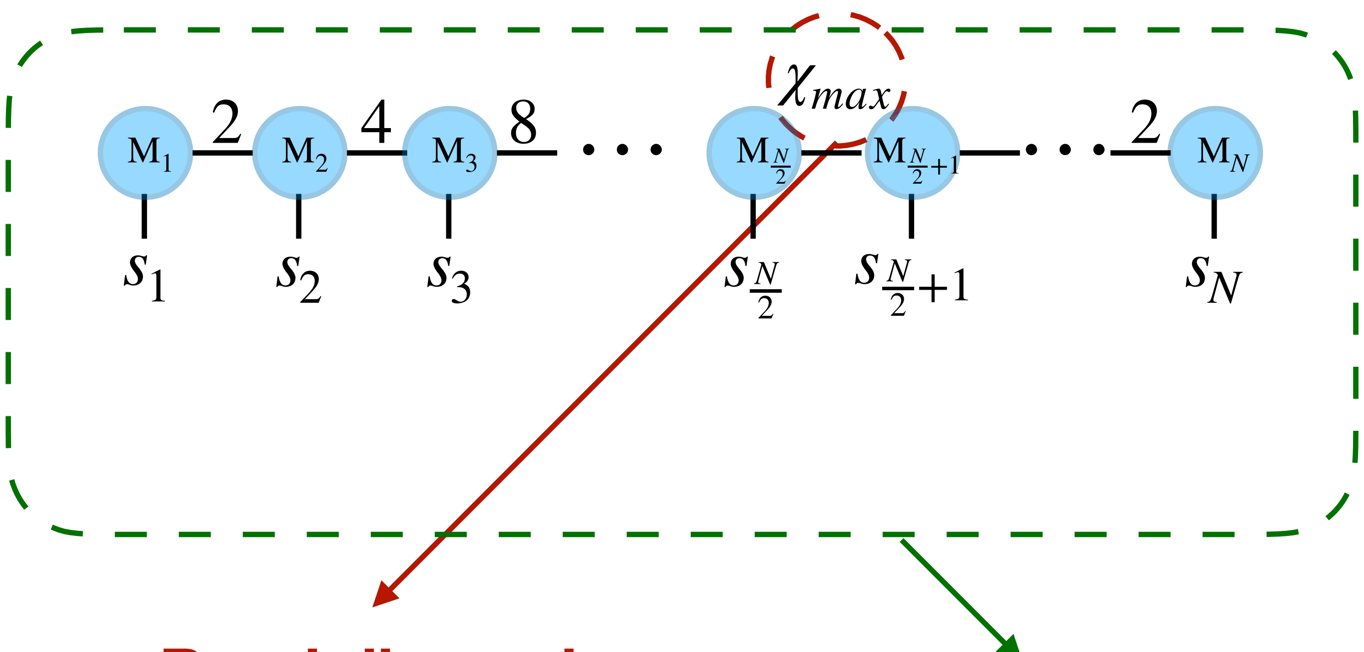
We keep the eigenvalues only if they are **big enough**. In this way, we are neglecting the sub-leading term for the state description.

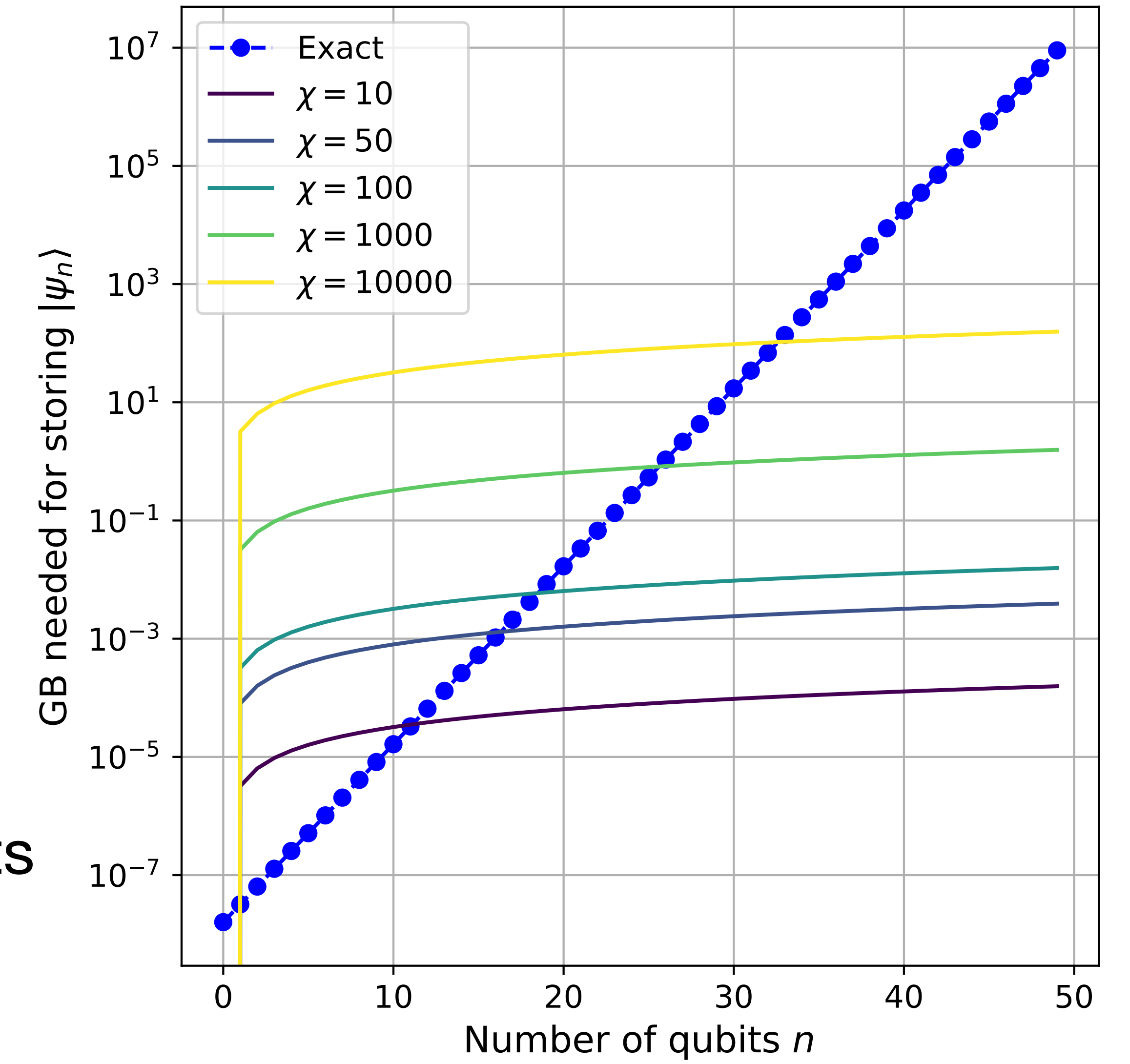
We keep only the **first highest** χ_{max} eigenvalues. In this way, we keep the quantum state manageable even for big number of qubits. However, this may be a strong approximation.

Matrix product states

The Matrix Product State representation of a quantum state is particularly efficient, due to the clever truncation.

- The truncation means that our tensors has **at most** dimensions $\chi_{max} \times \chi_{max} \times 2$.





Bond dimension

It controls the entanglement of the system

Number of coefficients **scales as**

$O\left(nd\chi^2\right)$

State Evolution

However, we have seen how to write an MPS starting from a state-vector. If we are not able to write the state-vector, due to RAM bounds, we cannot write the MPS?
The answer is no, and it is indeed what the simulator does.

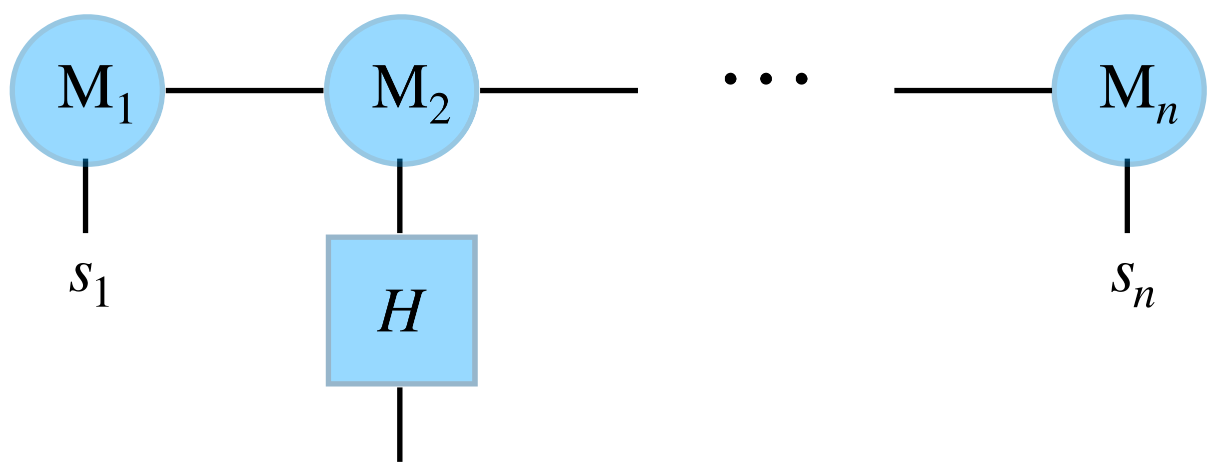
- We start by the state $|00\ldots0\rangle$. It is the usual starting state in quantum computation. Furthermore, being a **separable** state, which means with **no entanglement**, it can be described exactly by MPS with a bond dimension $\chi = 1$.

- We then apply gates to evolve the state, bringing it into the target state $|\psi\rangle$, as we would do normally with a quantum circuit.

- However, we introduce two limitations:
  - We can only apply 1-qubit and 2-qubits gates;
  - We can only work with quantum circuits with a **linear topology**;

One-qubit gates

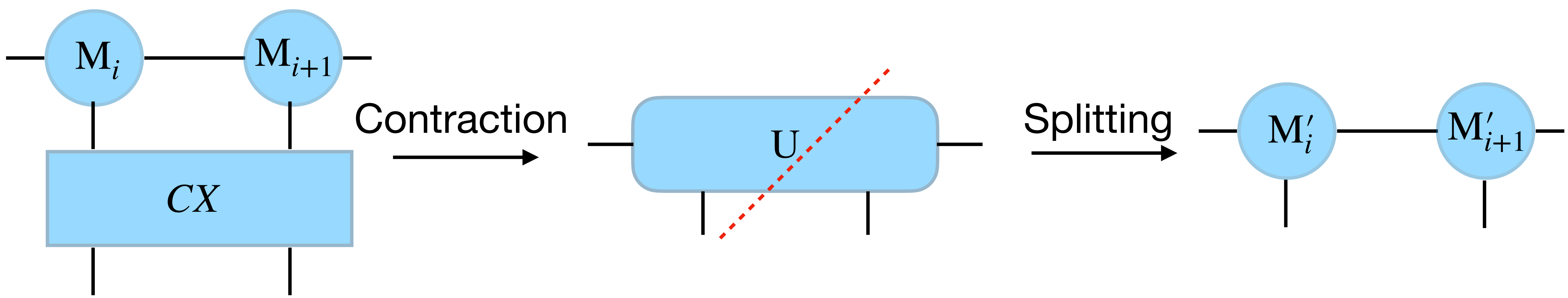
- Application of one-quit gates is easy, we simply have to contract the qubit tensor with the gate matrix:



- They do not introduce entanglement in the system, and thus do not change the bond dimension χ.

Two-qubit gates

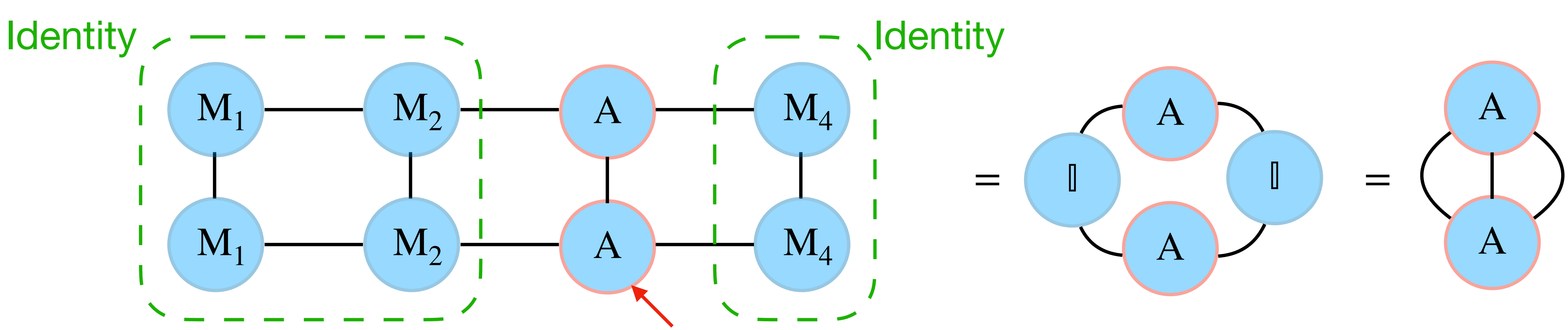
- Application of two-qubit gates is a little more involved, but we have all the ingredients to do it.
  - First, we need to reshape the gate matrix in an order-4 tensor.
  - Then, we perform the contraction.
  - Finally, we separate the tensors back.



- They introduce entanglement in the system, and thus the bond dimension χ might increase after the application of a two-bit gate, up to χ_{max}.

Orthogonality Center

- There are, however, some subtleties. The truncation induces an error, and we want to **minimise** that error.
- To do so, we have to set the **orthogonality center** of the tensor network.
- Very formal definition for the orthogonality center A. Check in the references if interested!
- Practically, A is a center of orthogonality if all the other tensors in the network are unitary, and so contract to the identity with their adjoint.



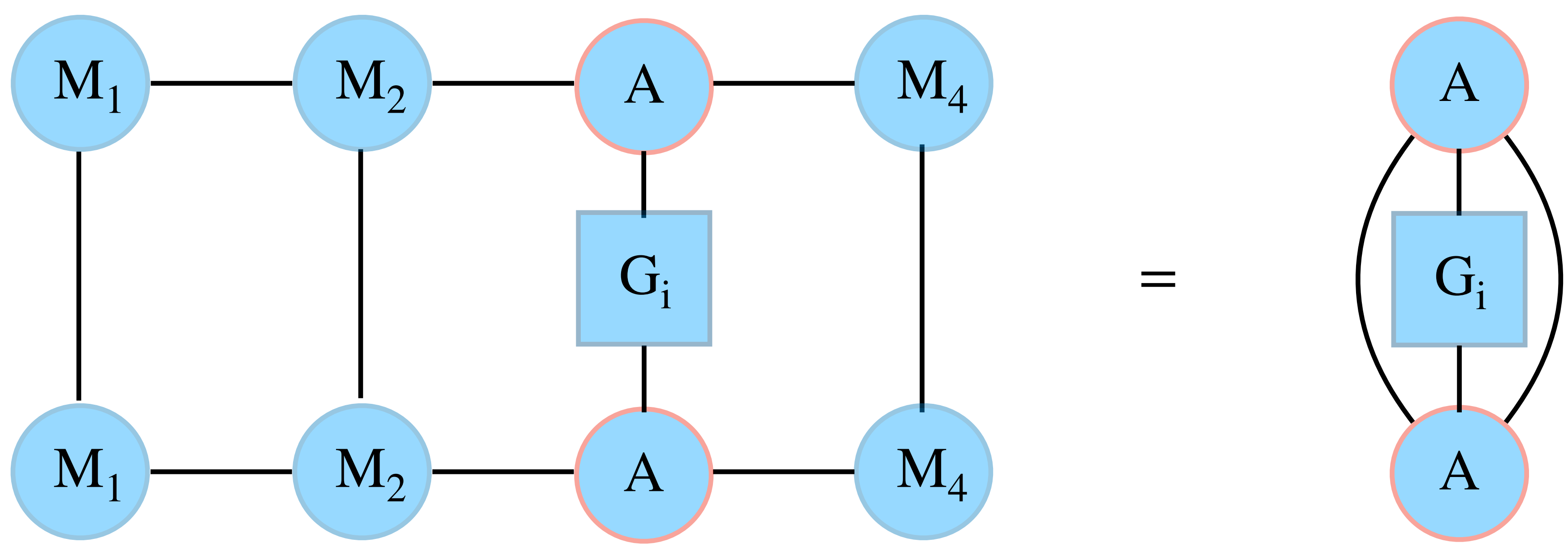
Orthogonality center
outlined in red

Operations on MPS

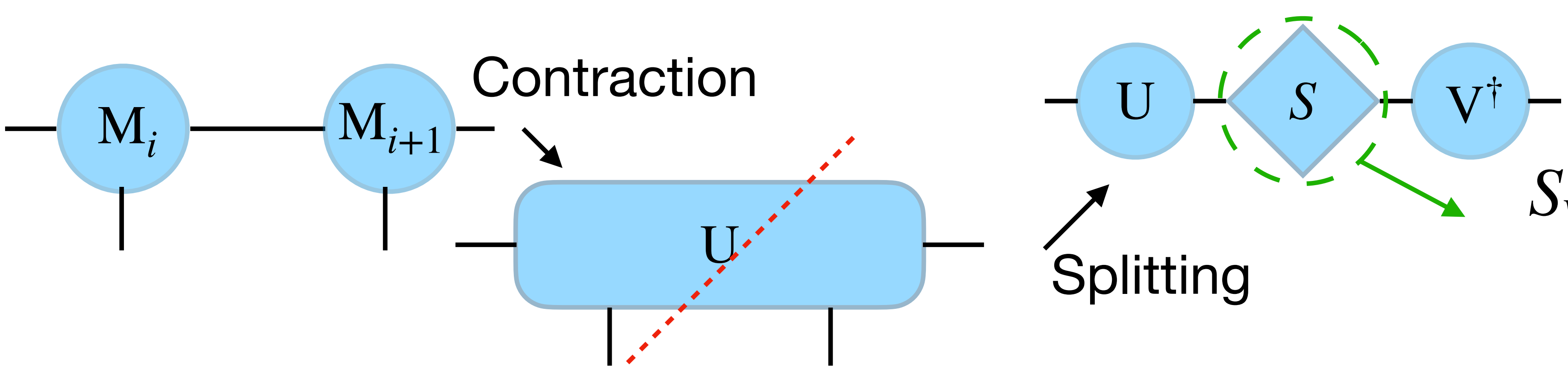
MPS are not only an efficient way of simulating quantum circuits. We can also **measure** interesting quantities:

- The expectation value of any single-site observable (gate):

$$\langle \psi | G_i | \psi \rangle =$$



- The entanglement entropy between two partition of the system:



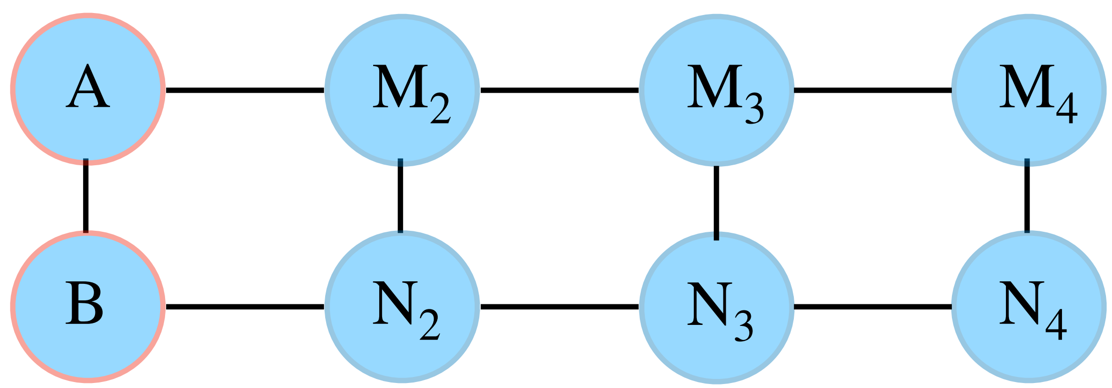
$$S_V = \sum_{i=1}^{min(\chi_{max},N)} s_i^2 \ln s_i^2$$

Operations on MPS

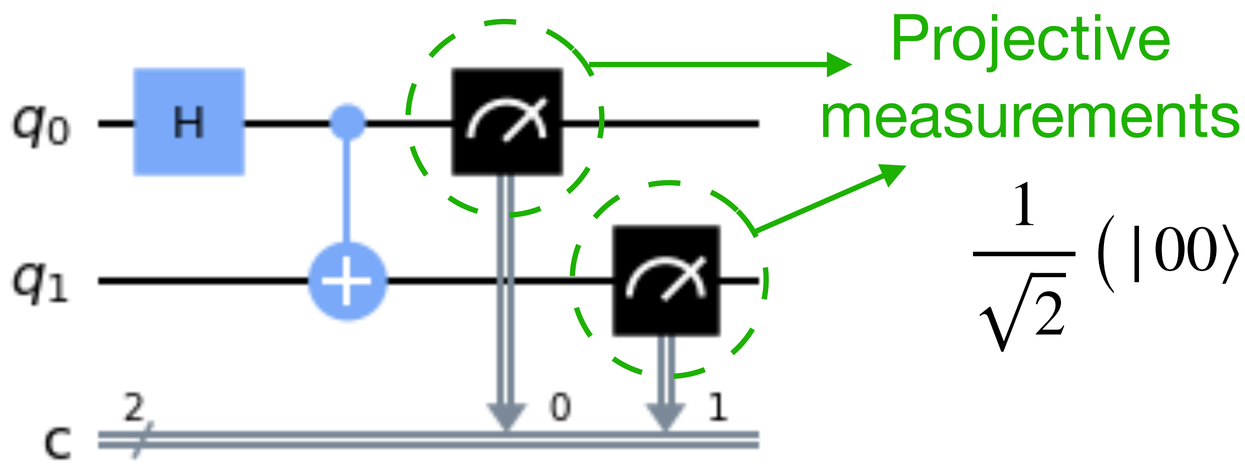
MPS are not only an efficient way of simulating quantum circuits. We can also **measure** interesting quantities:

- Scalar product between quantum state

$$\langle \psi | \phi \rangle \quad =$$



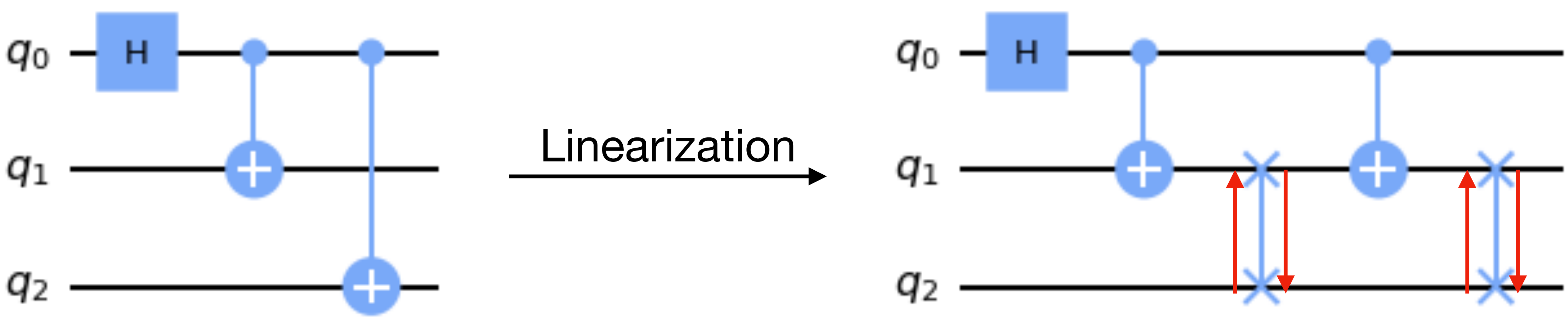
- Perform projective measurements



$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \longrightarrow \begin{cases} |00\rangle & \textbf{with prob } \frac{1}{2} \\ |11\rangle & \textbf{with prob } \frac{1}{2} \end{cases}$$

Circuit Linearisation

MPS are restrained to be used in a **linear topology**. However, any circuit can be mapped into a linear topology using **swap gates**.

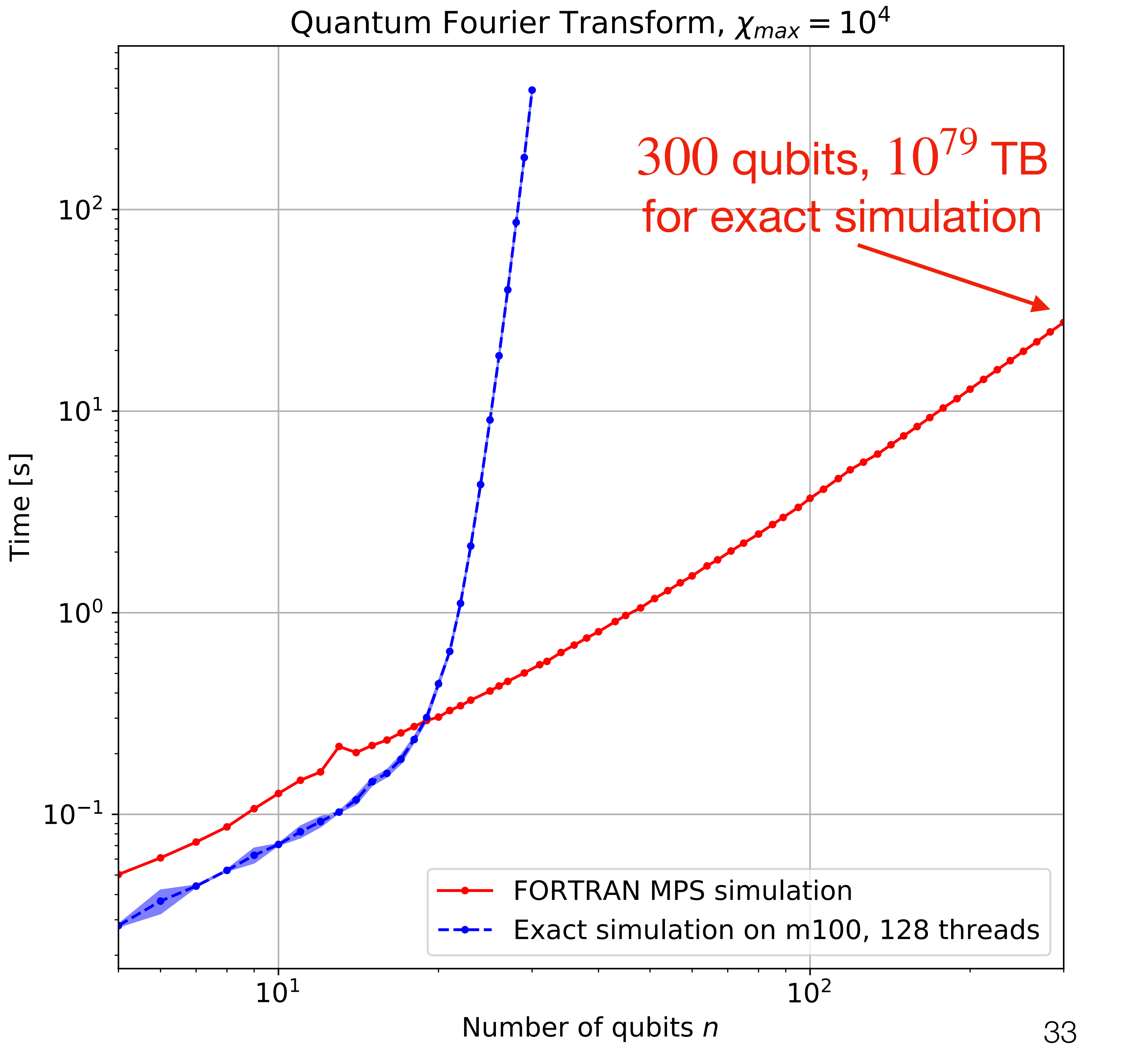


There are algorithms that **minimise** the number of swaps to map an arbitrary circuit to a linear topology.

QFT on m100

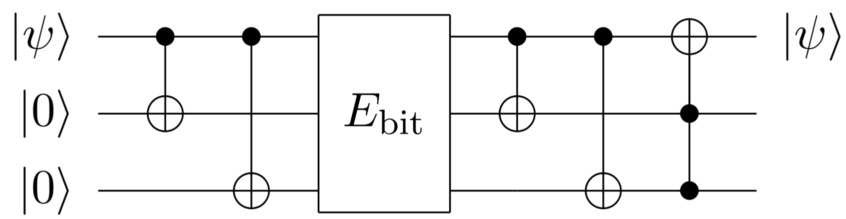
Marconi 100 Supercomputer
Nodes: 980
Cores: 32/node
RAM: 256 GB/node

Image from CINECA



Future development

Quantum Supremacy



Quantum Machine Learning

QUANTUM COMPUTING

Quantum computational advantage using photons

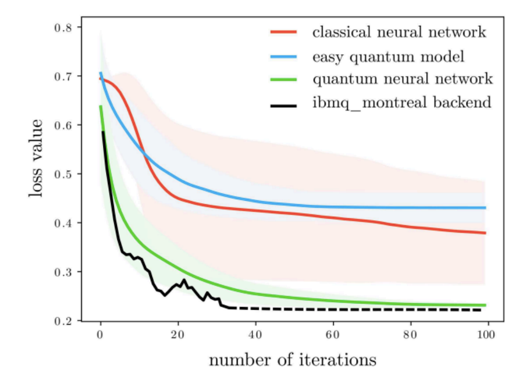
Han-Sen Zhong[1,2]*, Hui Wang[1,2]*, Yu-Hao Deng[1,2]*, Ming-Cheng Chen[1,2]*, Li-Chao Peng[1,2], Yi-Han Luo[1,2], Jian Qin[1,2], Dian Wu[1,2], Xing Ding[1,2], Yi Hu[1,2], Peng Hu[3], Xiao-Yan Yang[3], Wei-Jun Zhang[3], Hao Li[3], Yuxuan Li[4], Xiao Jiang[1,2], Lin Gan[4], Guangwen Yang[4], Lixing You[3], Zhen Wang[3], Li Li[1,2], Nai-Le Liu[1,2], Chao-Yang Lu[1,2]†, Jian-Wei Pan[1,2]†

Zhong, Han-Sen, et al.
"Quantum computational advantage using photons."
Science 370.6523 (2020): 1460-1463.

The power of quantum neural networks

Amira Abbas[1,2], David Sutter[1], Christa Zoufal[1,3], Aurelien Lucchi[3], Alessio Figalli[3], and Stefan Woerner[1,*]

[1] *IBM Quantum, IBM Research – Zurich*
[2] *University of KwaZulu-Natal, Durban*
[3] *ETH Zurich*



Fault-tolerant threshold

Abbas, Amira, et al.
"The power of quantum neural networks."
Nature Computational Science 1.6 (2021): 403-409.

Bibliography

1. Montangero, Simone. "Montangero, and Evenson, Introduction to Tensor Network Methods." (2018).
2. Biamonte, Jacob, and Ville Bergholm. "Tensor networks in a nutshell." *arXiv preprint arXiv:1708.00006* (2017).
3. Orús, Román. "A practical introduction to tensor networks: Matrix product states and projected entangled pair states." *Annals of physics* 349 (2014): 117-158.
4. Eisert, Jens. "Entanglement and tensor network states." *arXiv preprint arXiv:1308.3318* (2013).
5. Silvi, Pietro. "Tensor Networks: a quantum-information perspective on numerical renormalization groups." *arXiv preprint arXiv:1205.4198* (2012).
6. Paeckel, Sebastian, et al. "Time-evolution methods for matrix-product states." *Annals of Physics* 411 (2019): 167998.
7. https://www.tensors.net/

Final effort!

Try it yourself!

- Open the jupyter notebook "mps.ipynb"
- Play around! Use all the functions you need, try new functions or whatever!
- All the time left (- the time of a cup of coffee)

If you have questions or suggestion on the lesson:

- Ask them aloud, I will answer
- Ask them through Mentimeter, I will answer
- Write the suggestions on Mentimeter, I will try to improve

https://www.menti.com/zwnvkncnvx

Thank you
for your attention