Workshop sul calcolo nell'INFN

The QuantumTEA Cloud Platform

M. Ballarin (Unipd), L. Zangrando (INFN)

Loano 24/05/2023

QuantumTEA definition

Quantum Tensor-Network **E**mulator **A**pplications

- Tensor network emulators for quantum systems;
- Quantum computer emulator powered by tensor networks;
- Emulates complex quantum systems “via quantum circuits” with a large number of qubits O(100);
- Run general-purpose quantum circuits (algorithms).

Quantum Computing and Simulation Center

QUANTUM COMPUTING AND SIMULATION CENTER

- QuantumTEA is developed in the context of QCSC project;
- World Class Research Infrastructures (WCRI) project of University of Padua;
- Objectives of QCSC:
  - Establish one of the first general-purpose quantum computer in Italy at the Department of Physics and Astronomy of the University of Padua;
  - Create a competence center to guide and support the development and the inclusion of quantum technologies in the Italian academic and business environment.

QCSC partners

QUANTUM COMPUTING AND SIMULATION CENTER

Università degli Studi di Padova

Università degli Studi dell'Aquila

INFN Istituto Nazionale di Fisica Nucleare

TQT

Padua QTech

Università di Pavia

CINECA

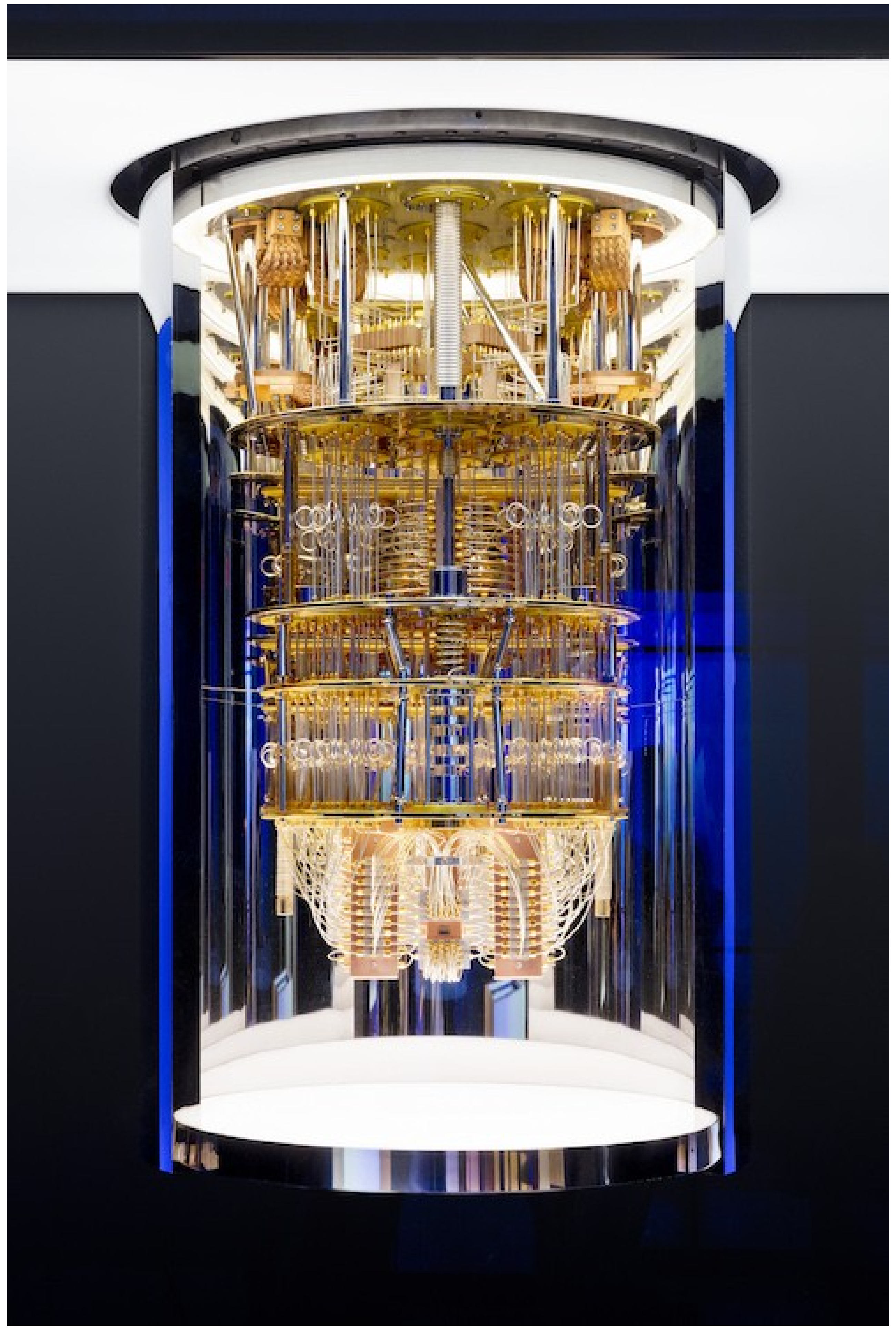
Q@TN

arakne do I.T. better

neat

Why develop a quantum emulator? (1/2)

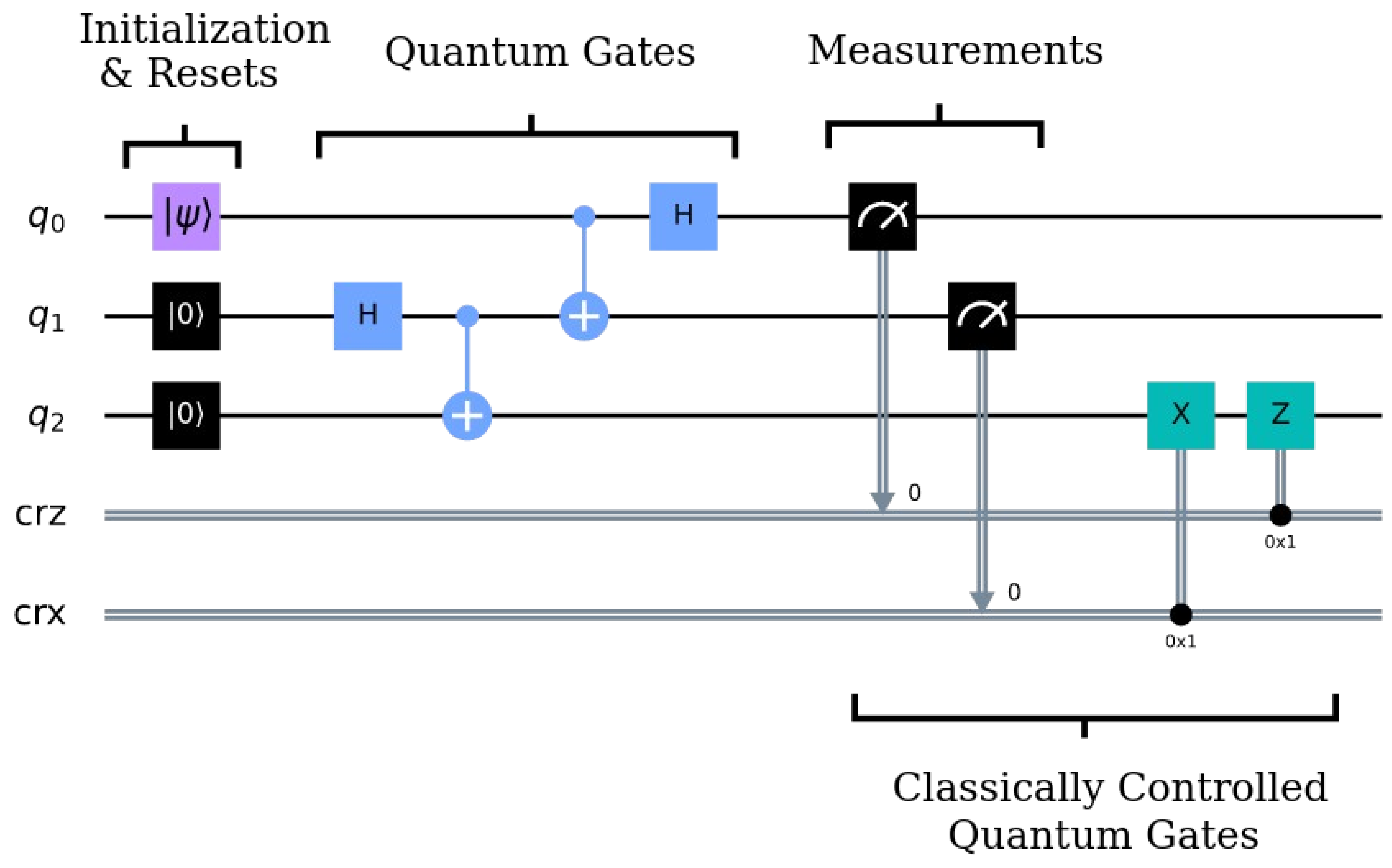
- Quantum computers are not production-ready yet;
- Today's quantum computers are:
  - very noisy systems with a limited number of qubits;
  - difficult to control and to isolate from the surrounding environment;
  - Large number of computation errors, impossible to correct with quantum error correction algorithms.



A view inside the IBM Quantum System One

Why develop a quantum emulator? (2/2)

- With emulators we can:
  - Validate the result of a real Quantum Computer (QC) computation;
  - Test and benchmark QC performances;
  - Understand when we really need a QC;
  - For industry: develop solutions based on quantum algorithms for the moment QPU take off and become scalable.



The Qubit

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

The Qubit

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

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

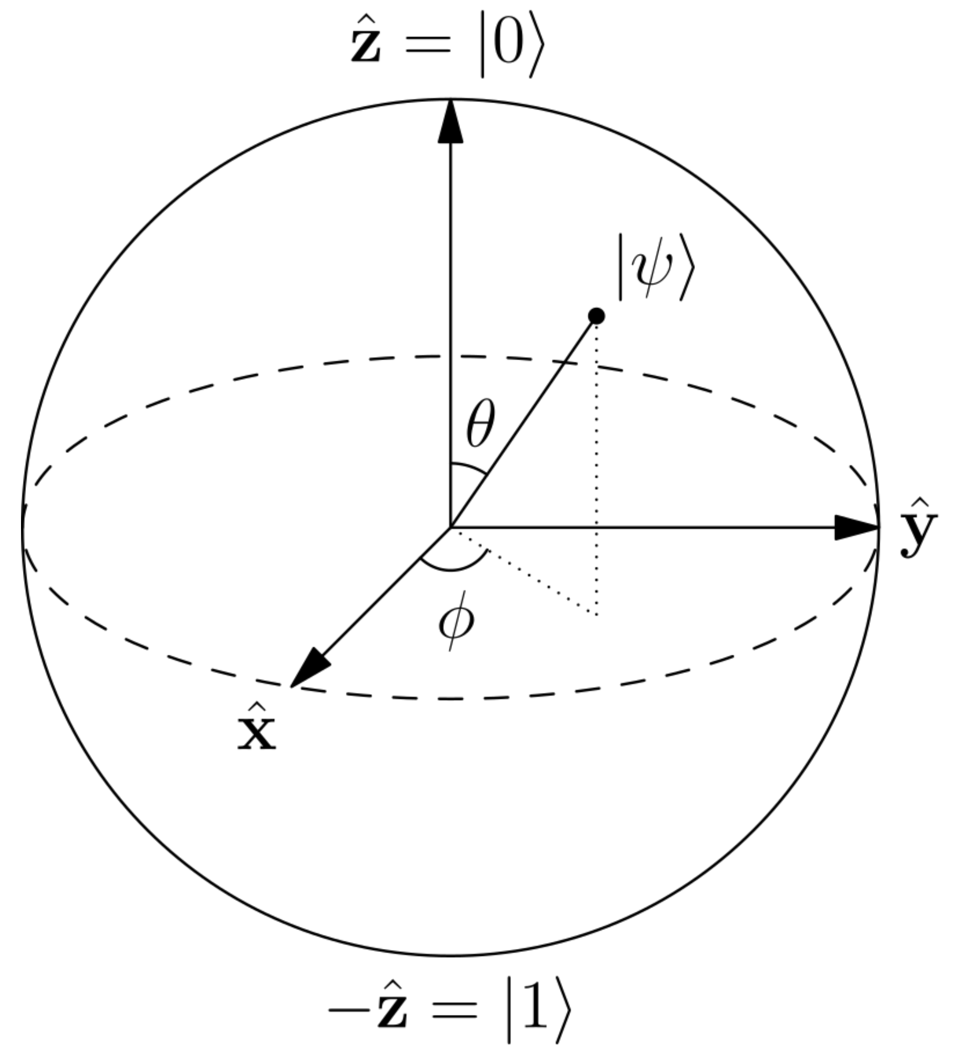
$$|\psi\rangle = \cos\theta|0\rangle + e^{i\phi}\sin\theta|1\rangle$$

The Qubit

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

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

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

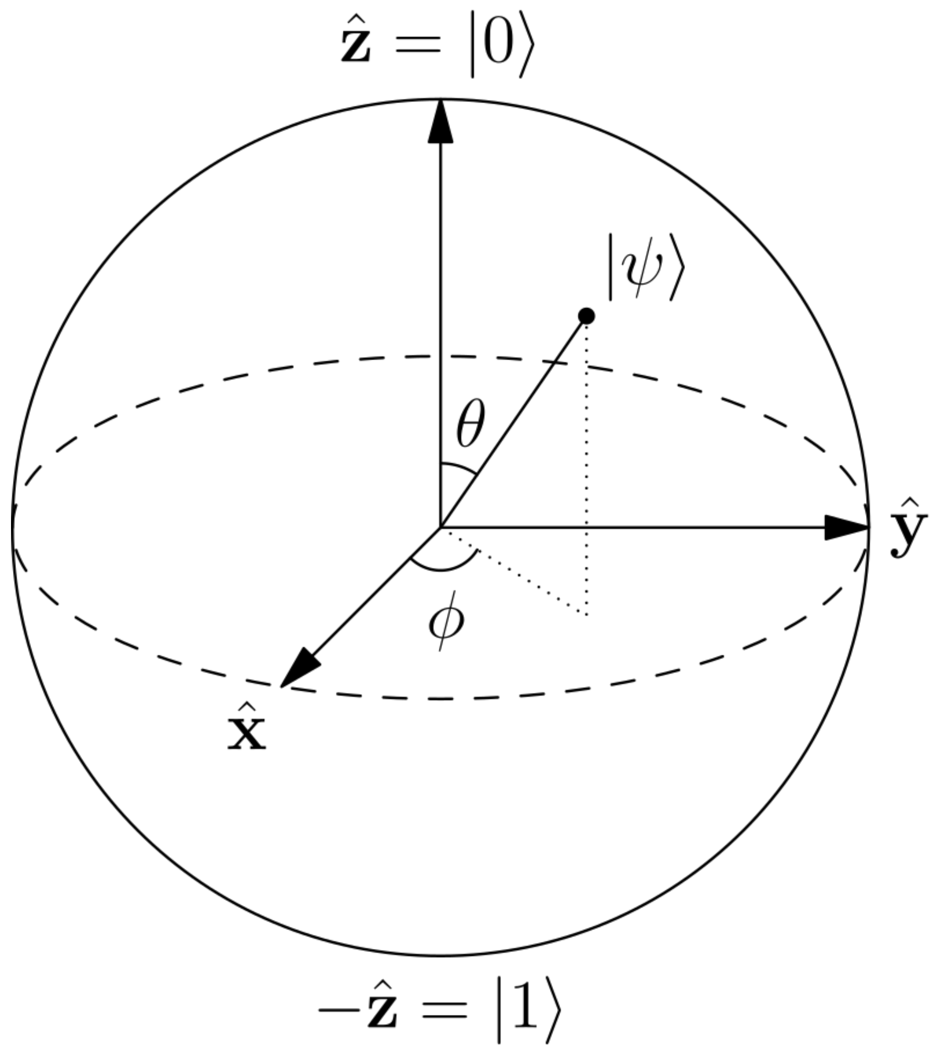


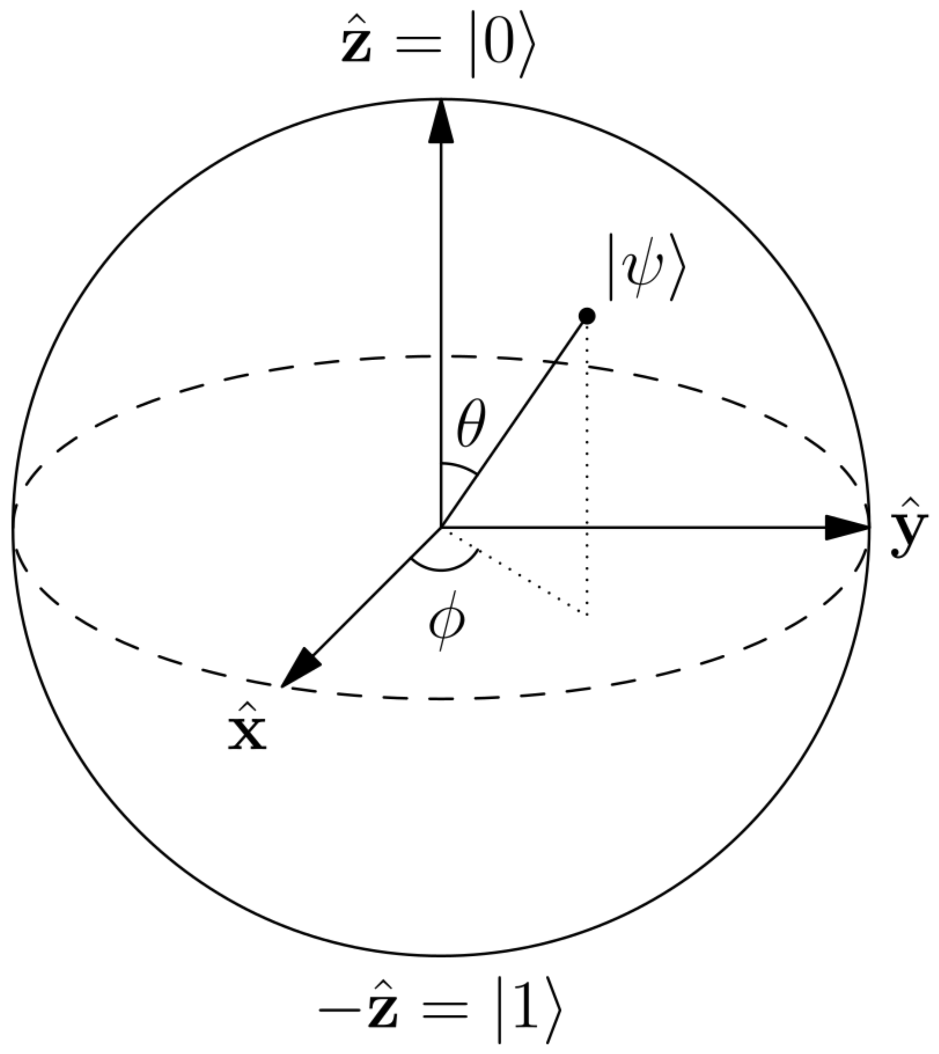
The Qubit

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

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

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



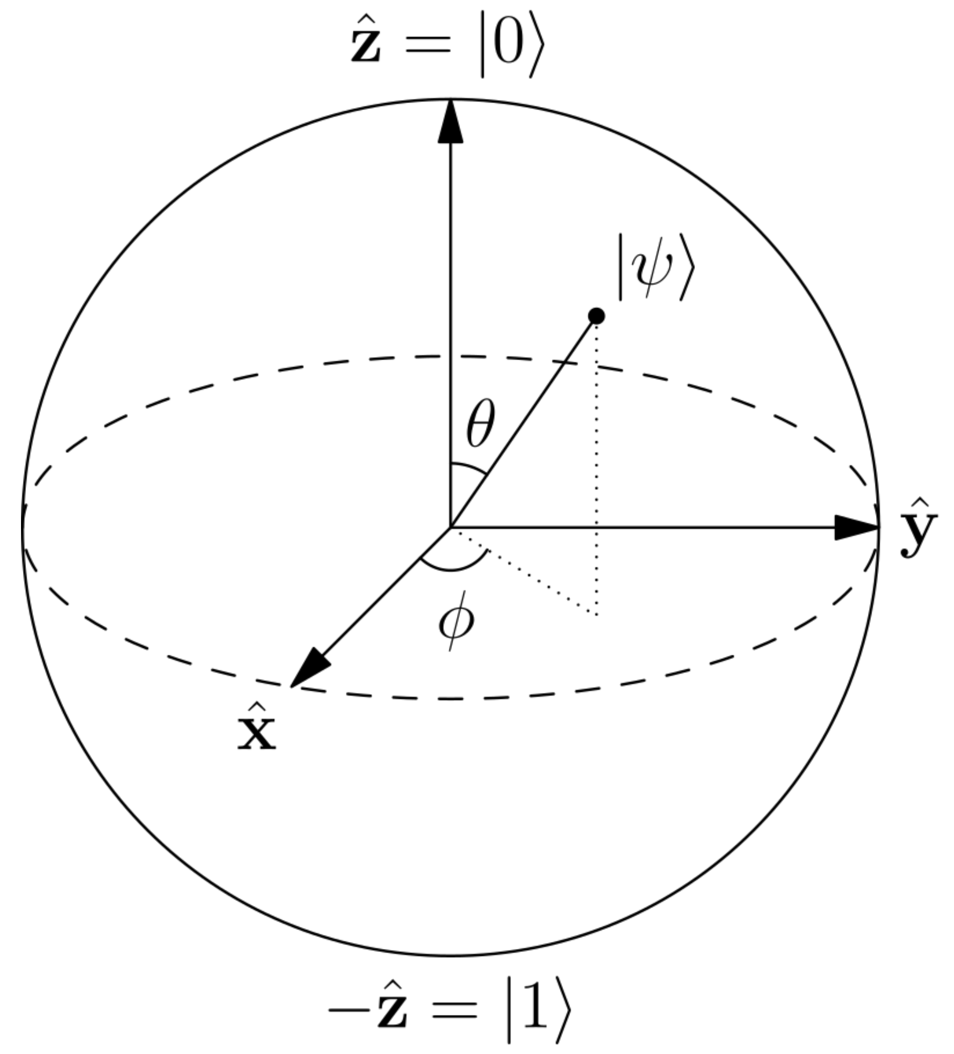


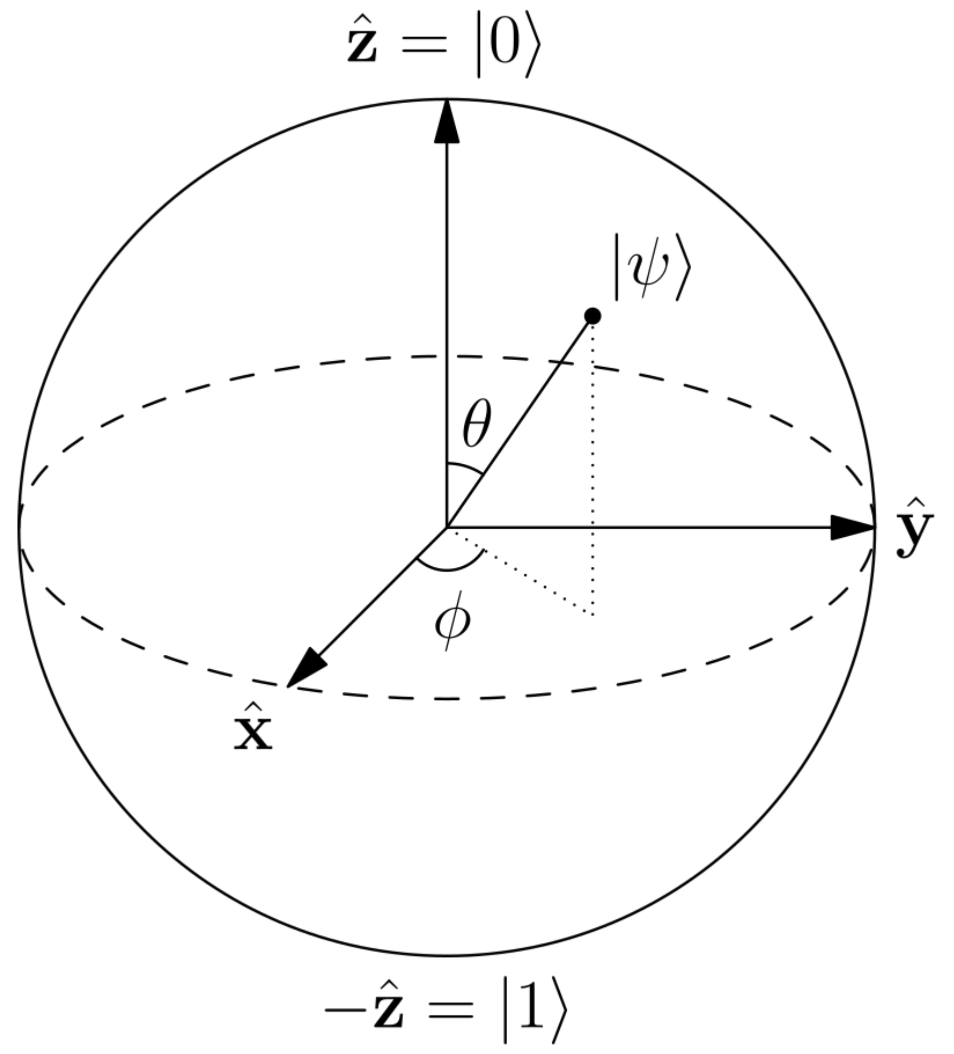
The Qubit

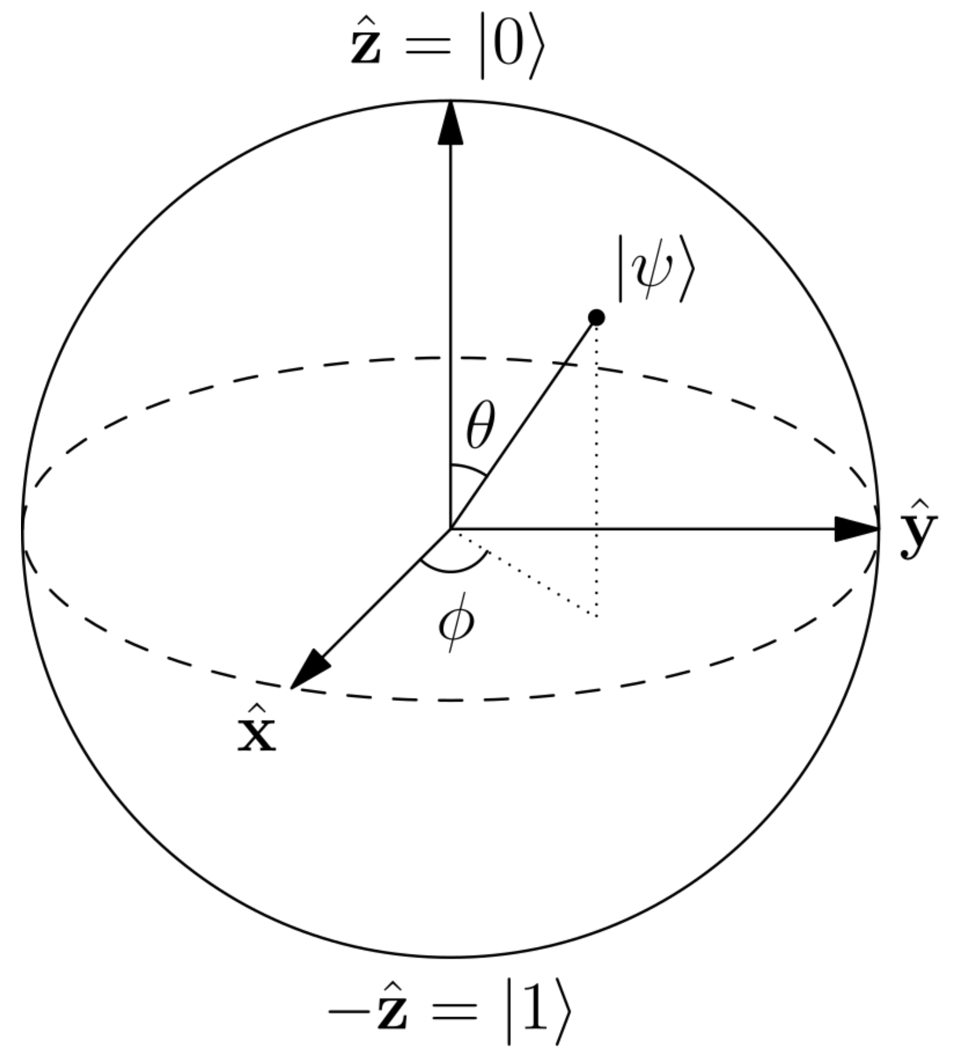
Classical bit $b \in \{0,1\}$

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

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





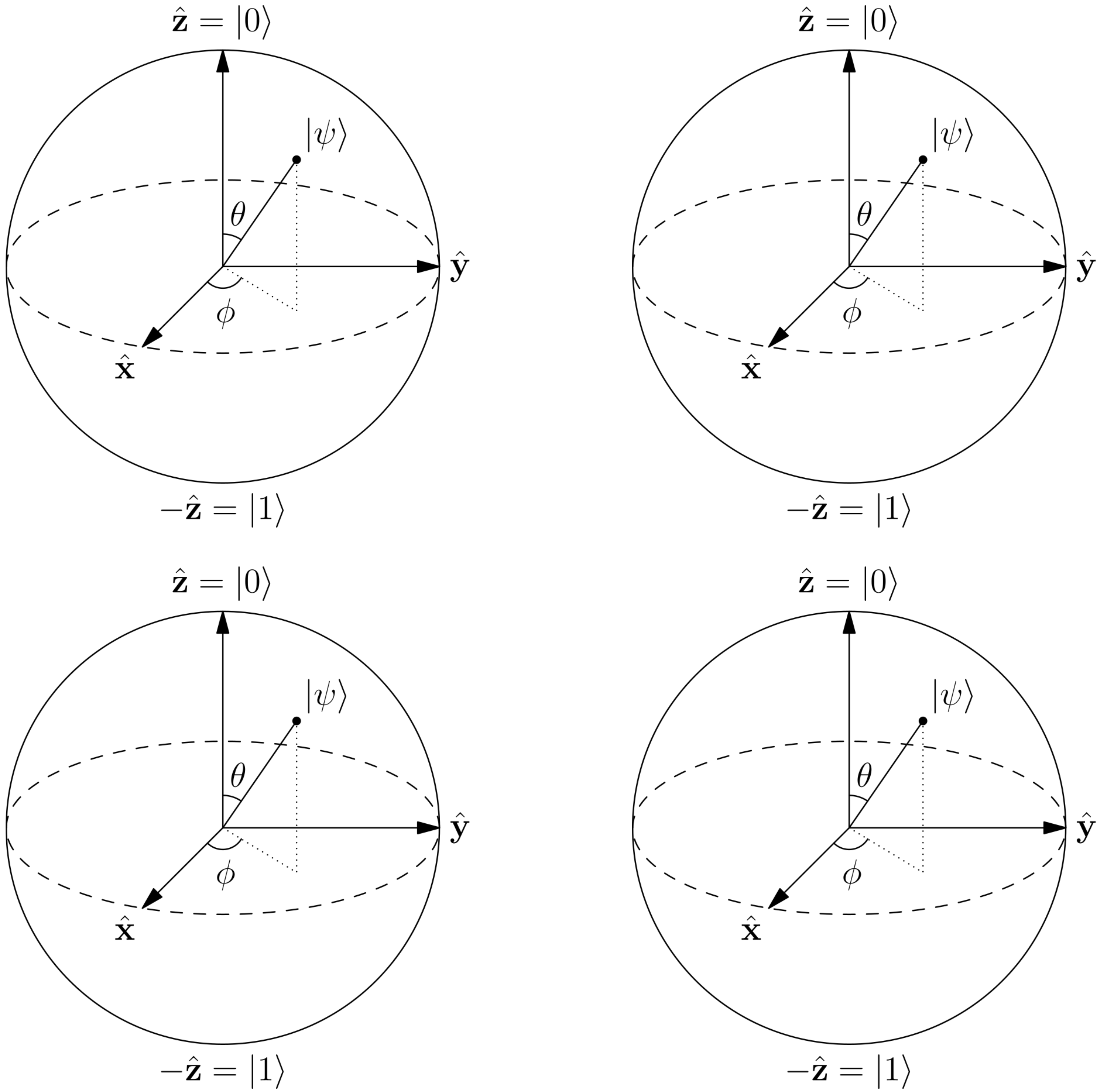


The Qubit

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

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

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

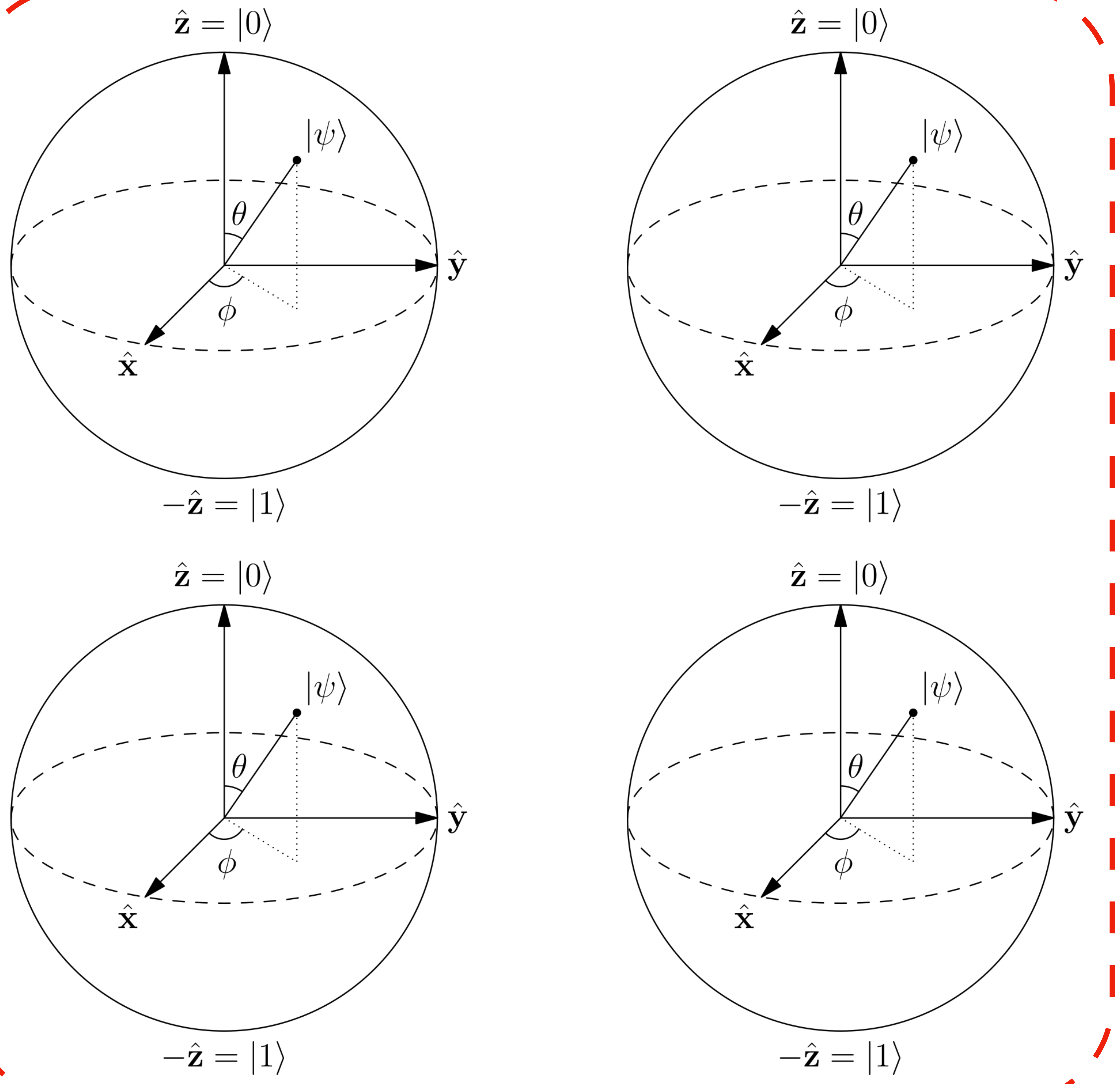


The Qubit

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

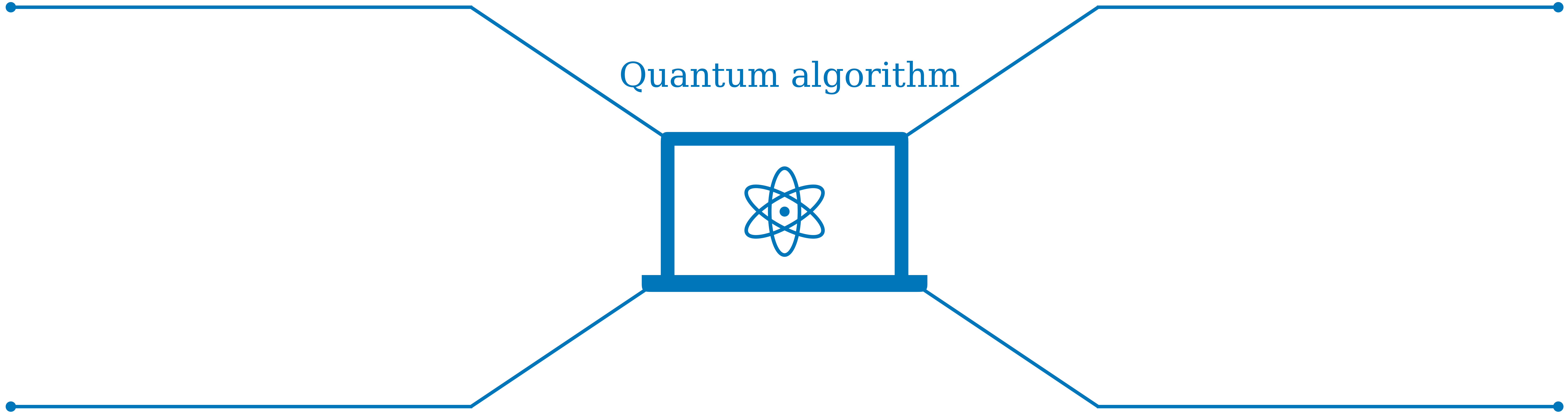
quantum qubit $|\psi\rangle \in \mathscr{H}, \dim(\mathscr{H}) = 2$

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



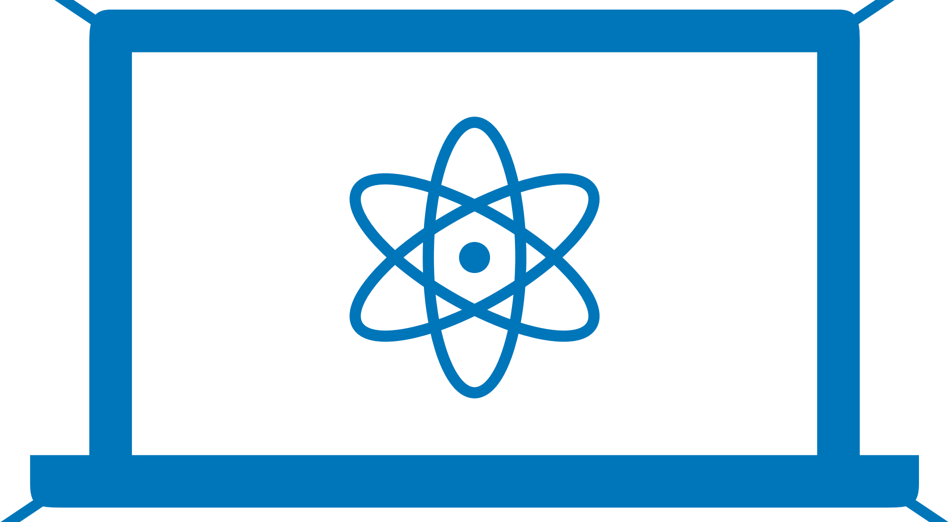
For 4 qubits you need a 2^4=16 coefficients.
For 50 qubits you would need 2^{50}=10^{15}

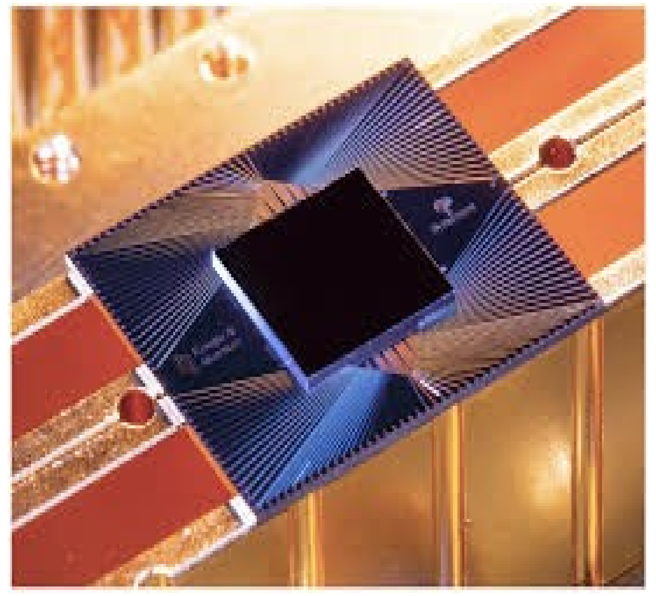
Running quantum algorithms



Running quantum algorithms

Quantum algorithm



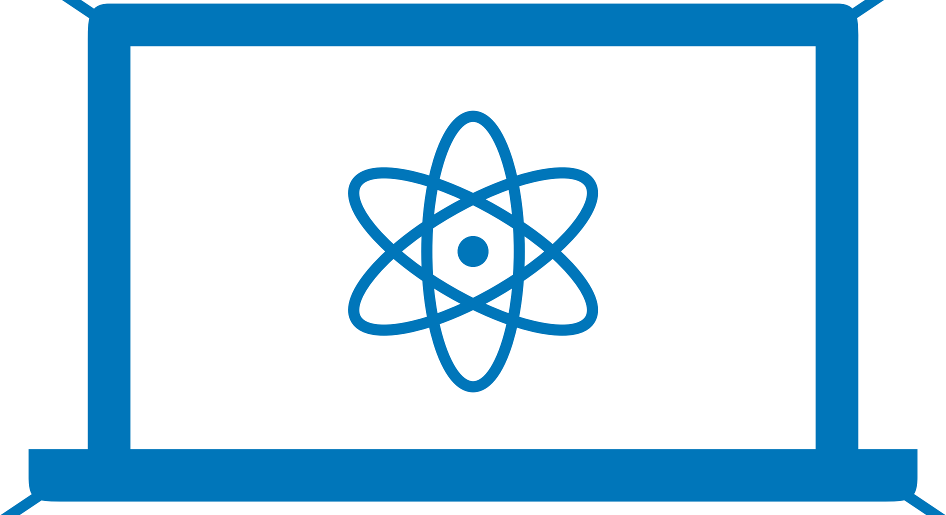


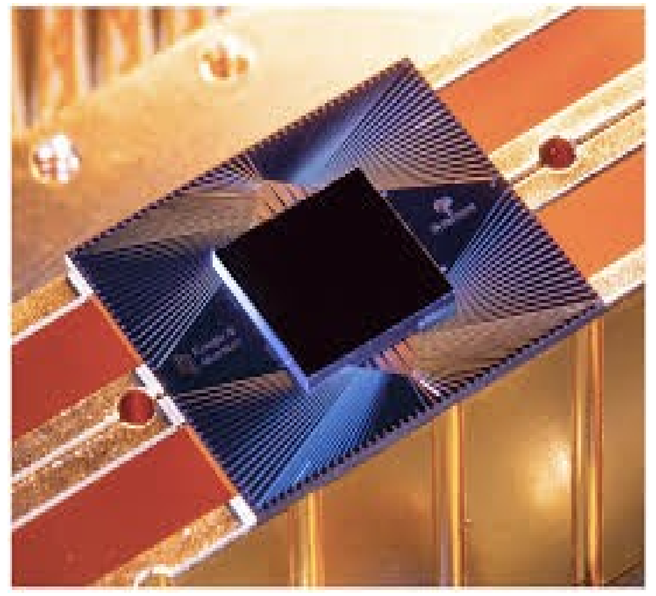
+ Real hardware
– Noisy
– Limited number of qubits

Quantum hardware

Running quantum algorithms

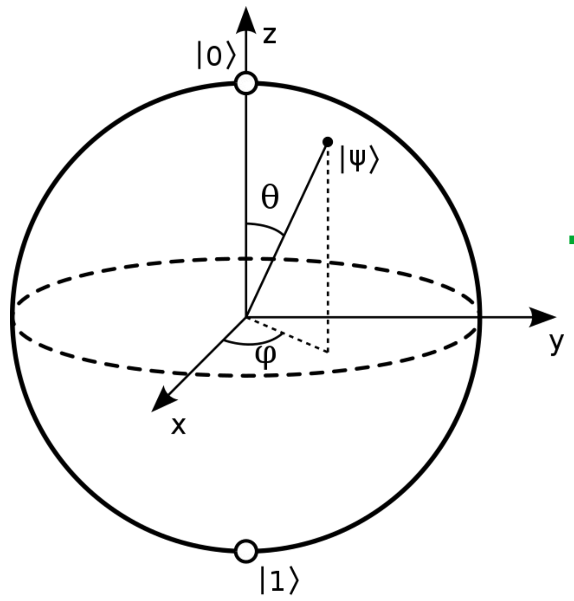
Quantum algorithm





+ Real hardware
– Noisy
– Limited number of qubits

Quantum hardware

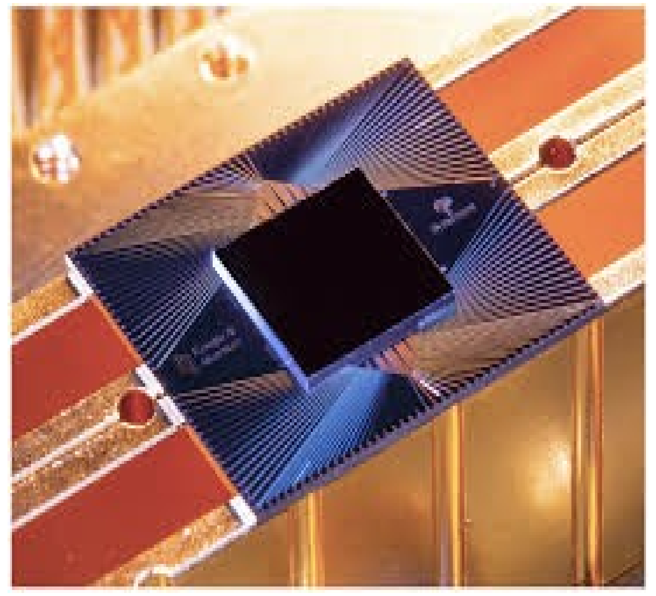


+ Access to exact state
– Limited number of qubits

Exact simulator

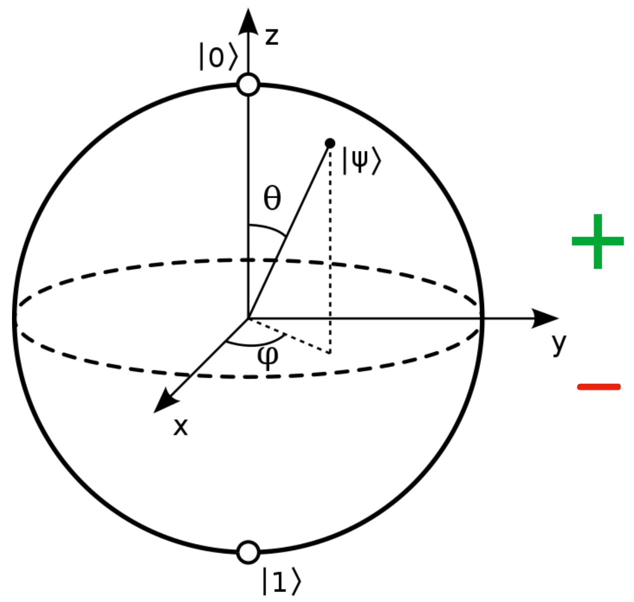
Running quantum algorithms

Quantum algorithm



+ Real hardware
– Noisy
– Limited number of qubits

Quantum hardware



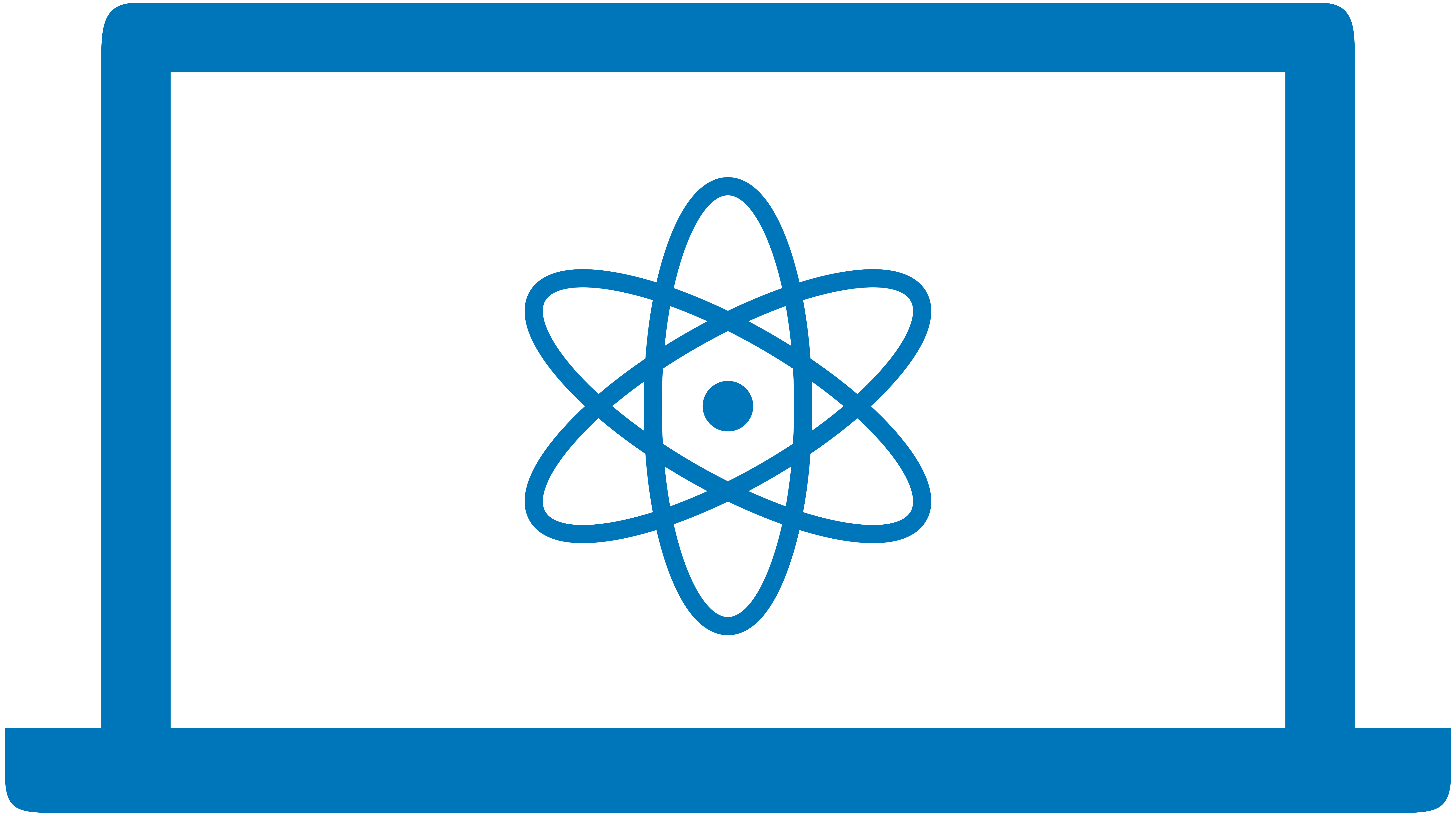
+ Access to exact state
– Limited number of qubits

Exact simulator

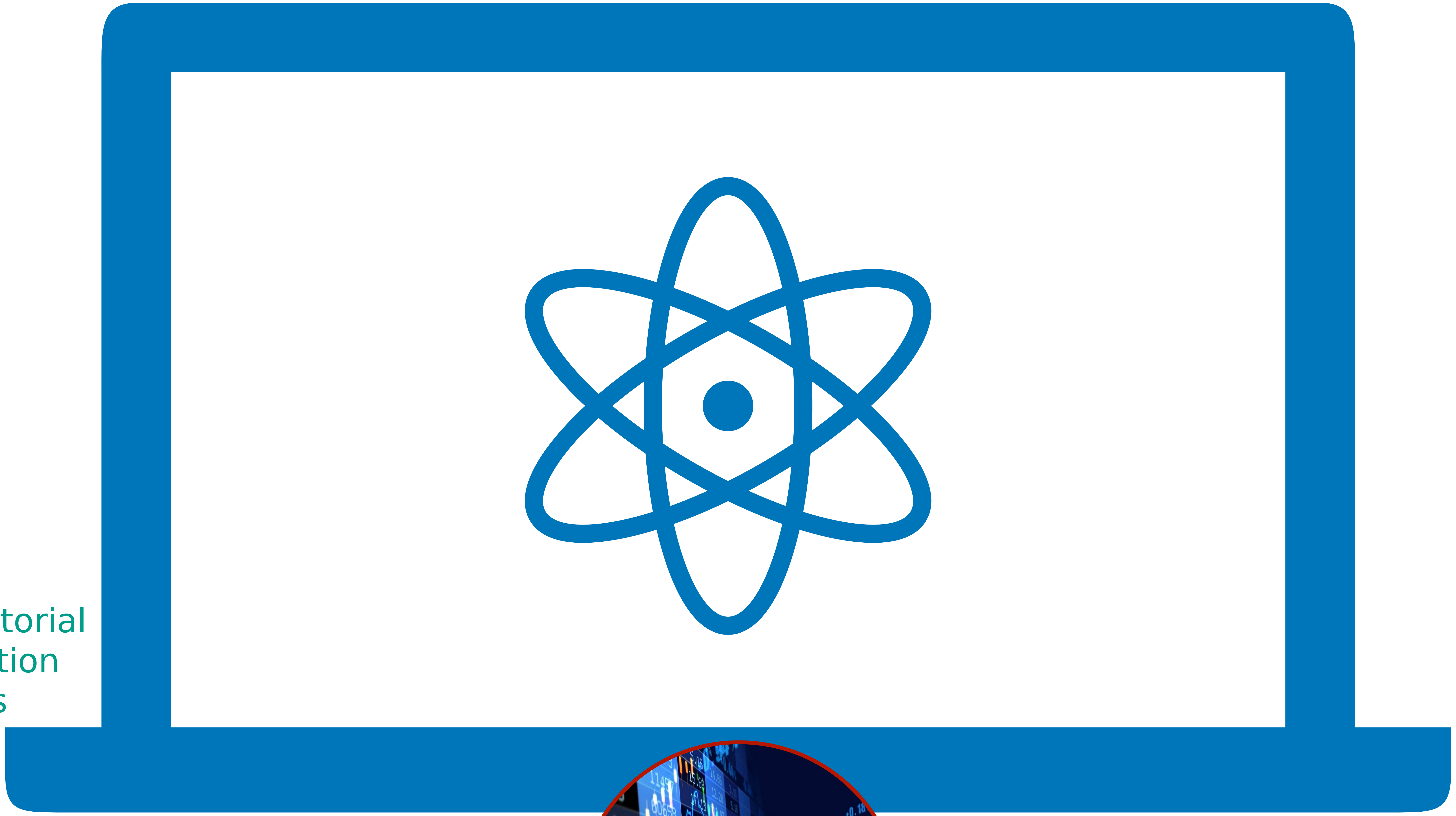
Tensor Network simulator

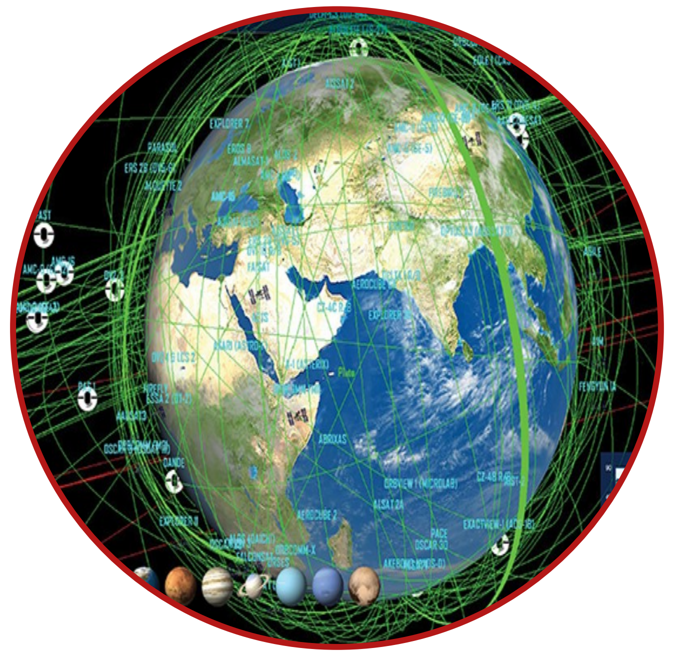
+ High number of qubits
– Limited entanglement

Quantum algorithms



Quantum algorithms





Earth
Observation

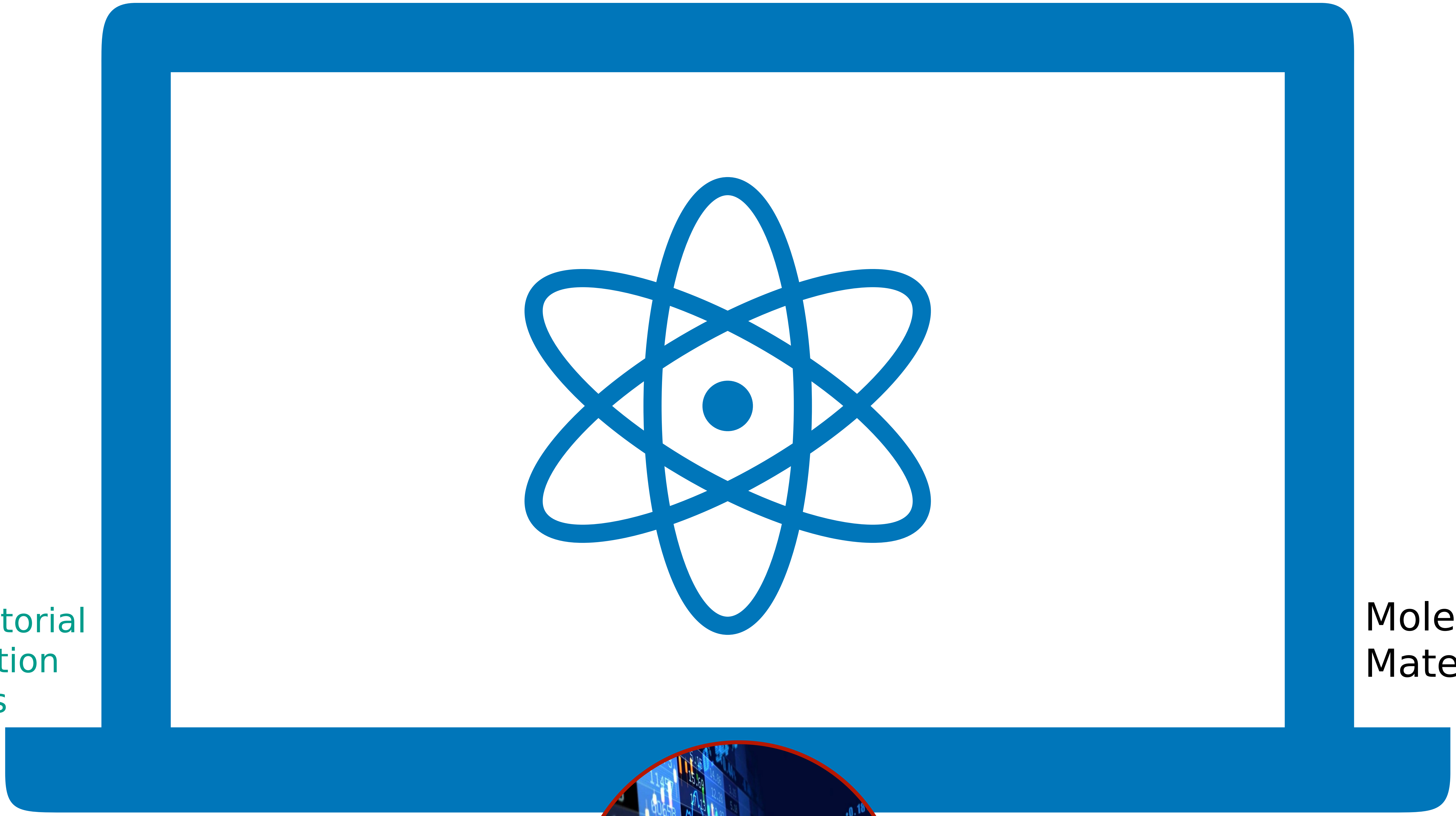
Combinatorial
optimization
problems

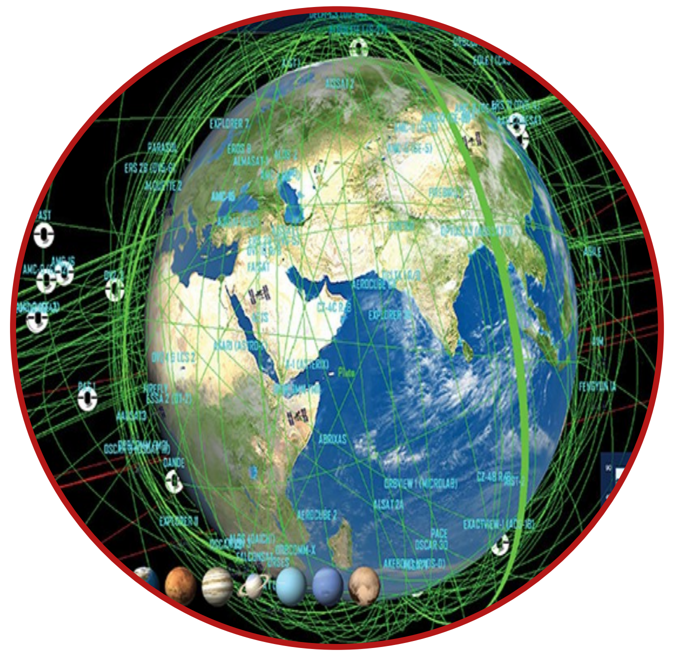
(QAOA, quantum
annealing, ...)

Traffic

Portfolio
optimization

Quantum algorithms





Earth
Observation

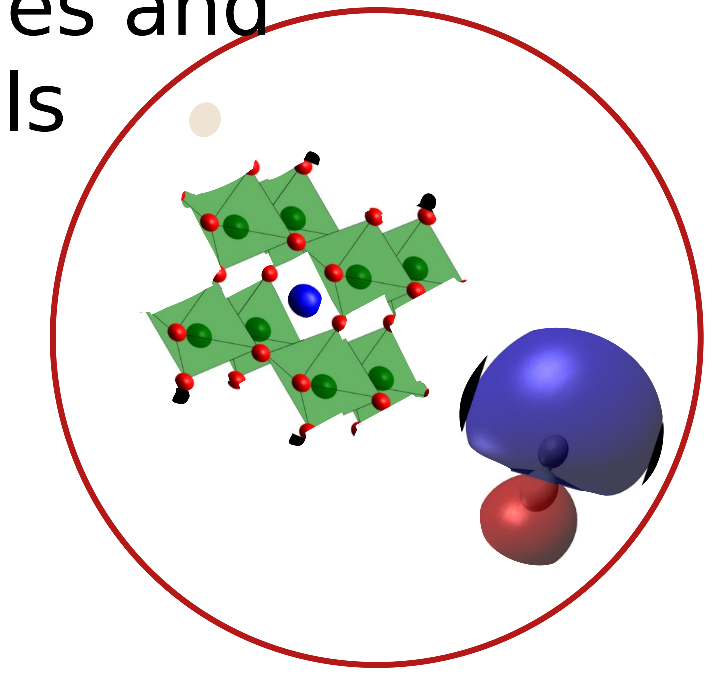
Traffic

Combinatorial
optimization
problems

(QAOA, quantum
annealing, ...)

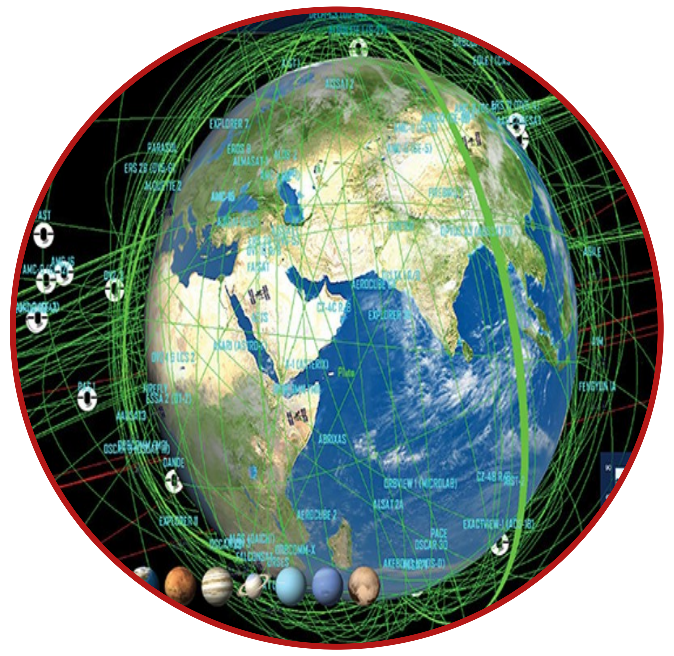
Portfolio
optimization

Molecules and
Materials



(VQE, quantum deflation, ...)

Quantum algorithms



Earth
Observation

Machine Learning

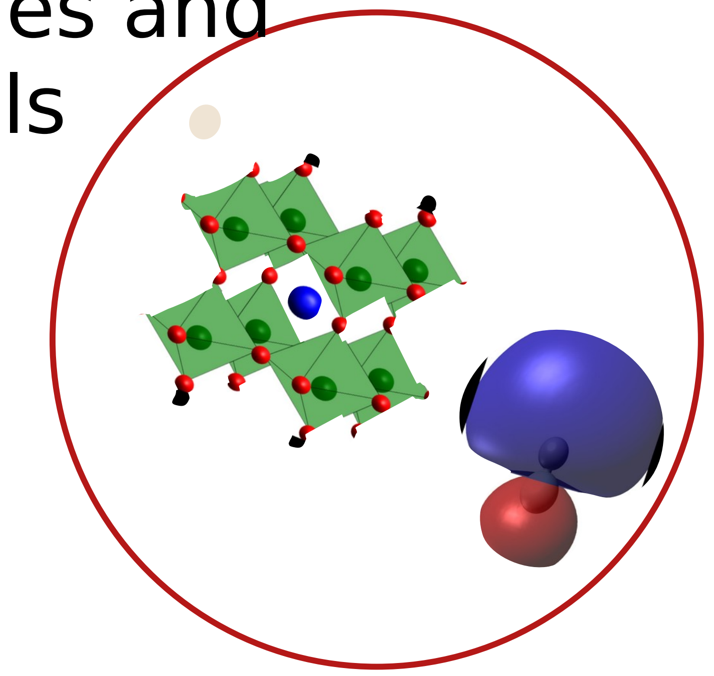
Traffic

Combinatorial
optimization
problems

(QAOA, quantum
annealing, ...)

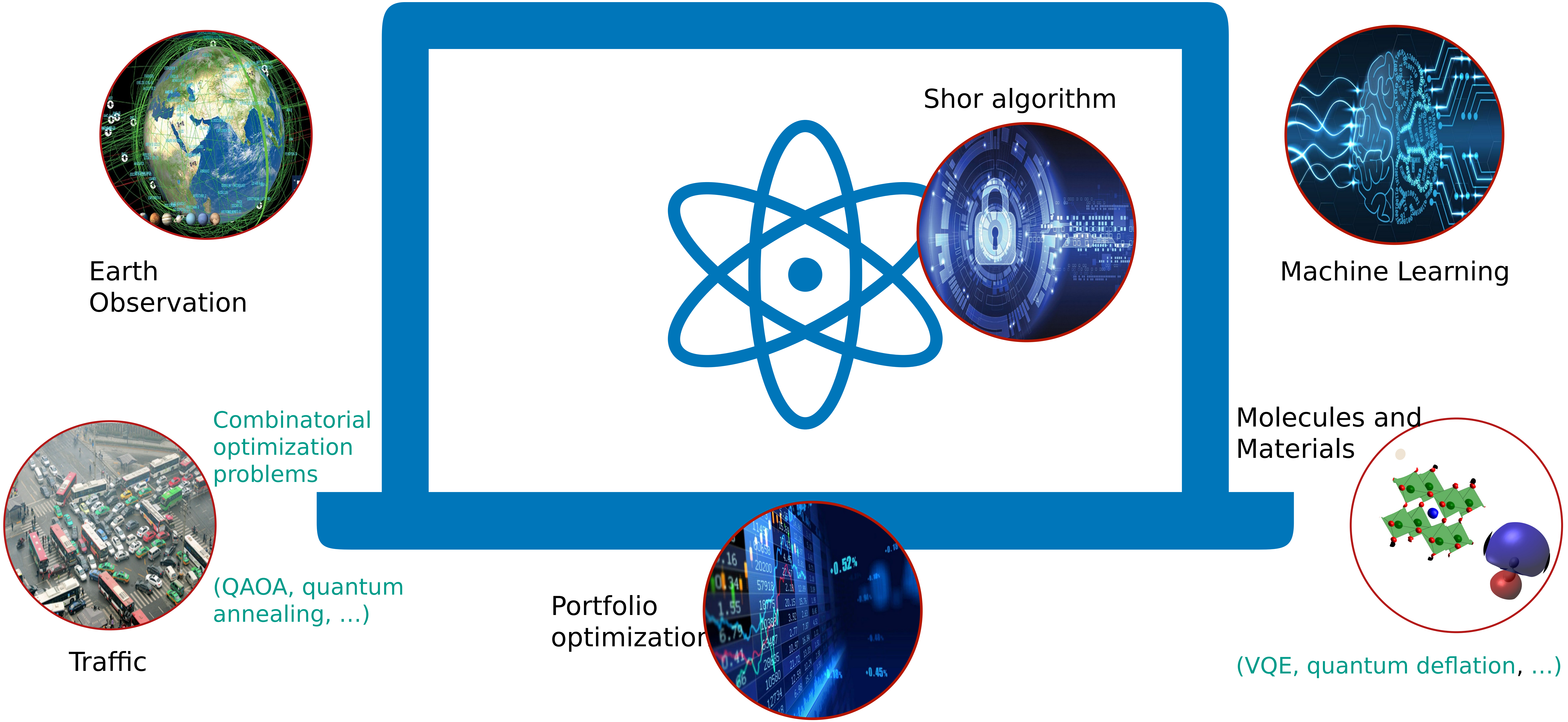
Portfolio
optimization

Molecules and
Materials

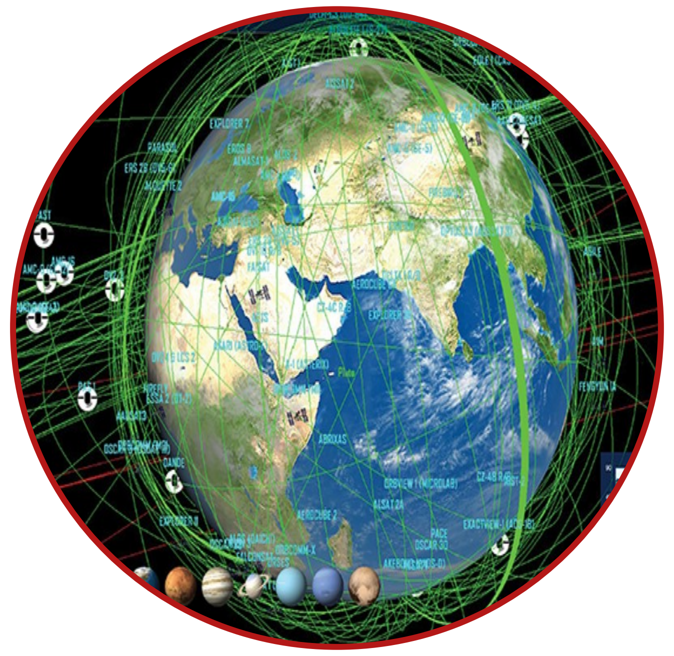


(VQE, quantum deflation, ...)

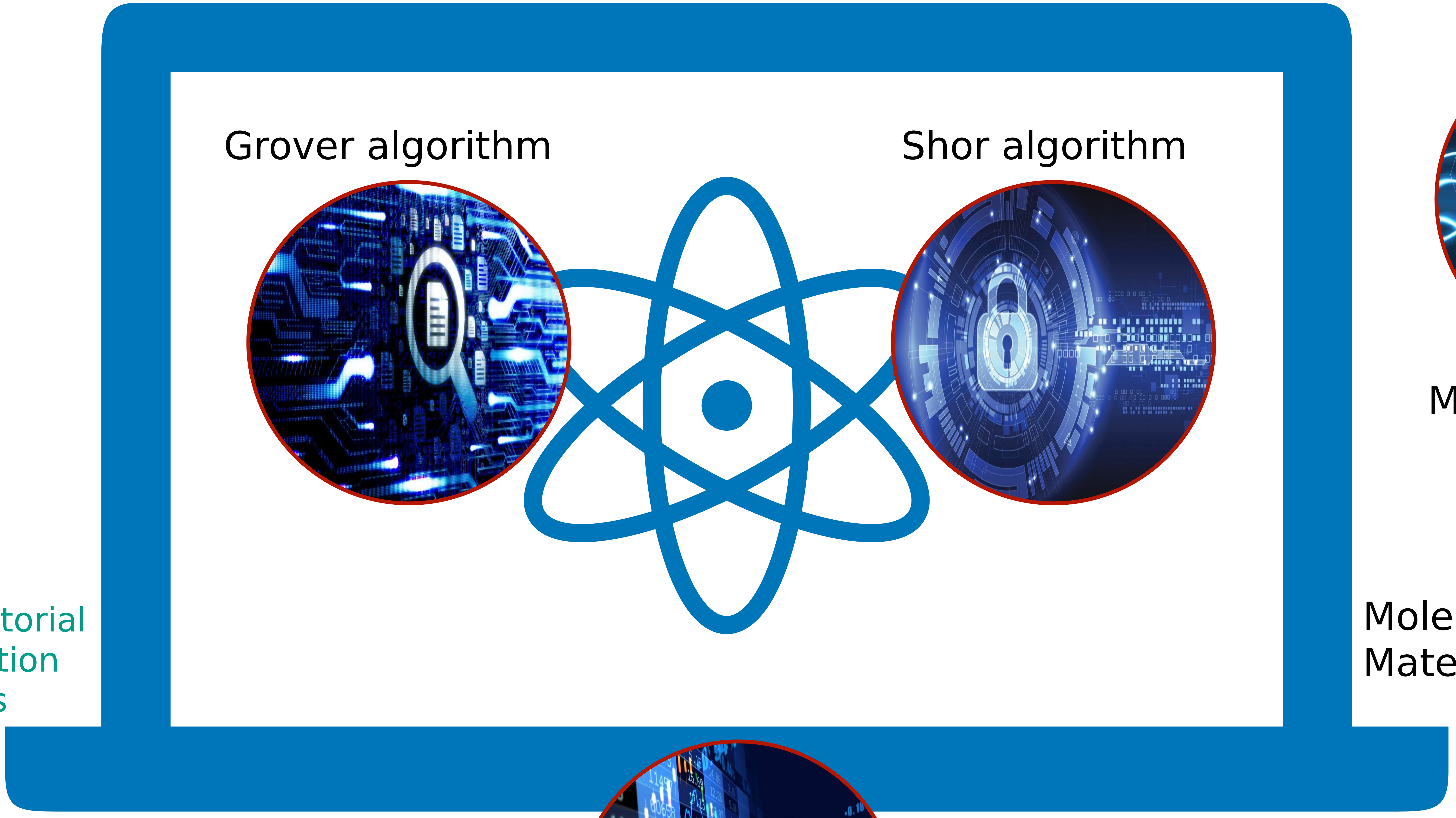
Quantum algorithms



Quantum algorithms



Earth Observation



Machine Learning

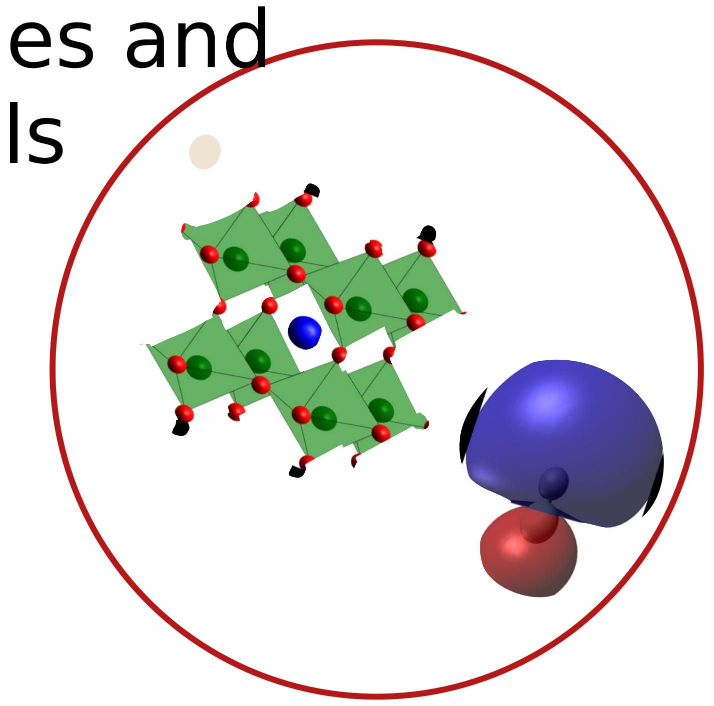
Combinatorial optimization problems

Traffic

(QAOA, quantum annealing, …)

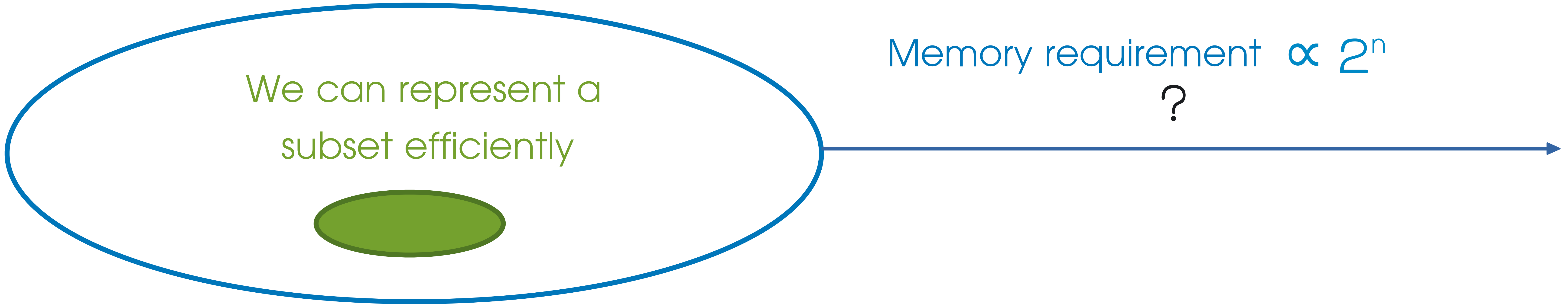
Portfolio optimization

Molecules and Materials

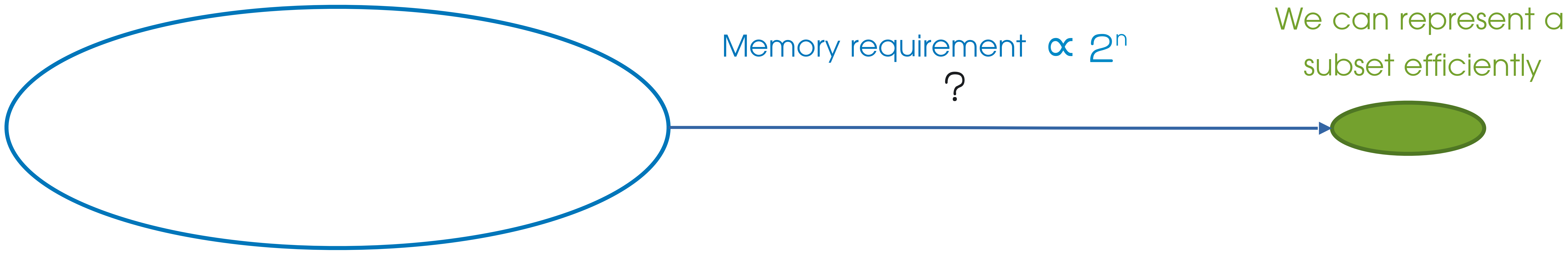


(VQE, quantum deflation, …)

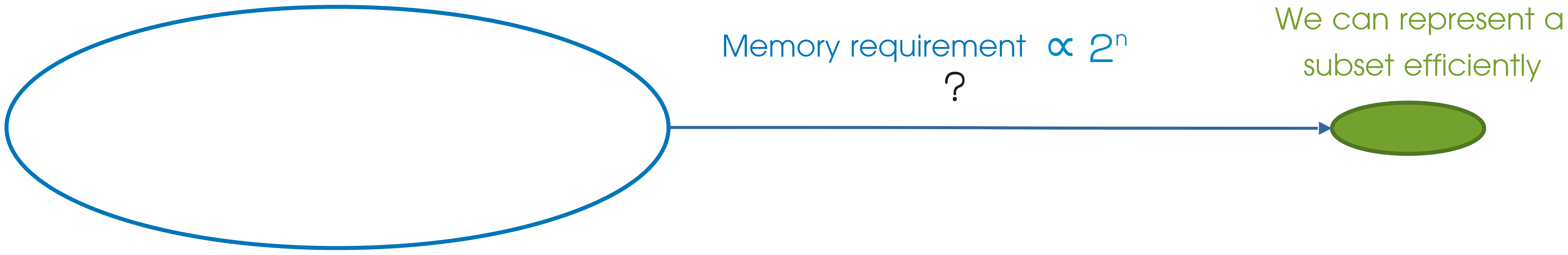
Entanglement and compression



Entanglement and compression



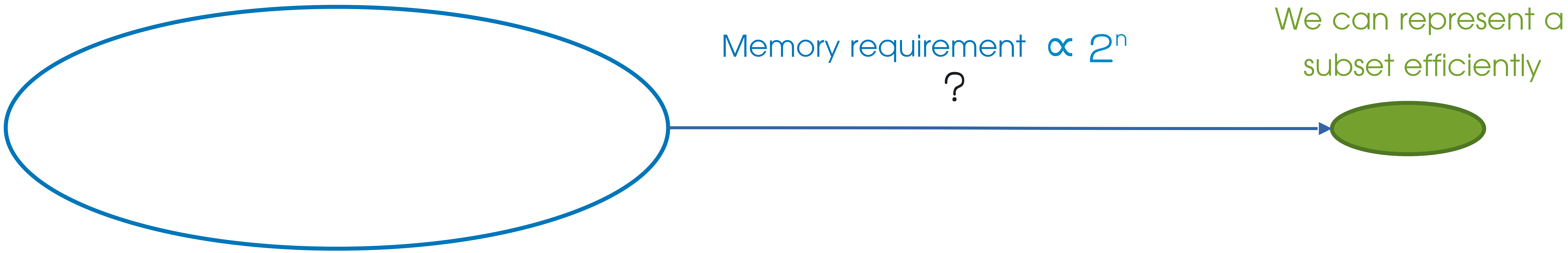
Entanglement and compression



Classical bit string

0000

Entanglement and compression

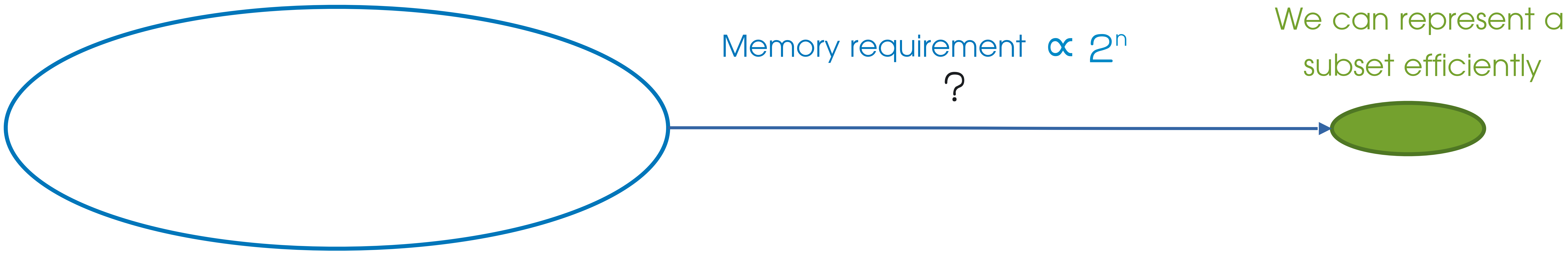


Classical bit string

0000

0011

Entanglement and compression



Classical bit string

0000

0011

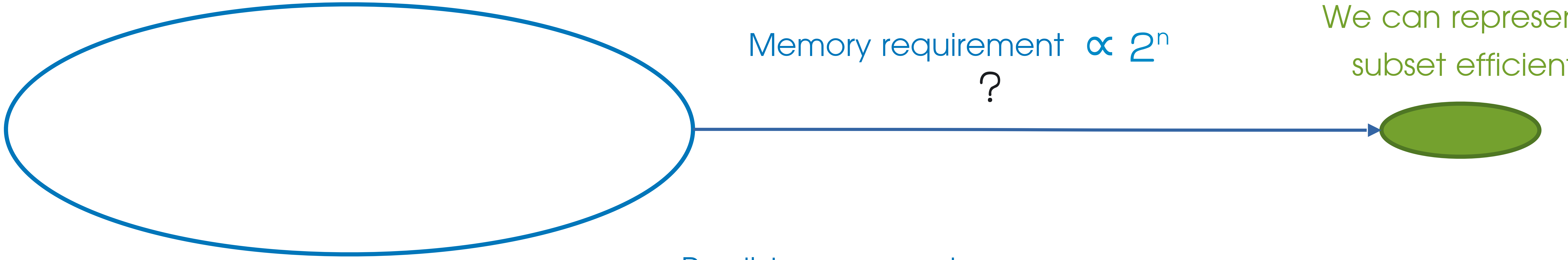
RANDOM

Entanglement and compression

Memory requirement $\propto 2^n$

?

We can represent a subset efficiently



Possible compression quantified by Shannon entropy

Classical bit string

0000

0011

RANDOM

Entanglement and compression

Memory requirement $\propto 2^n$

?

We can represent a subset efficiently

Possible compression quantified by Shannon entropy

Classical bit string	0000	0011	RANDOM
	Optimal Compression		
Quantum state	$\|0000\rangle$		

Entanglement and compression

Memory requirement $\propto 2^n$?

We can represent a subset efficiently

Possible compression quantified by Shannon entropy

Classical bit string	0000	0011	RANDOM
	Optimal Compression	Here we can compress something. The quantum state is **entangled**, it cannot be written as classical state	
Quantum state	$\vert 0000\rangle$	$\frac{1}{\sqrt{2}}\left(\vert 0000\rangle + \vert 1111\rangle\right)$	

Entanglement and compression

Memory requirement $\propto 2^n$?

We can represent a subset efficiently

Possible compression quantified by Shannon entropy

Classical bit string	0000	0011	RANDOM
	Optimal Compression	Here we can compress something. The quantum state is **entangled**, it cannot be written as classical state	No compression possible without approximations
Quantum state	$\|0000\rangle$	$\frac{1}{\sqrt{2}}(\|0000\rangle + \|1111\rangle)$	$\|RANDOM\rangle$

Entanglement and compression

Memory requirement $\propto 2^n$?

We can represent a subset efficiently

Possible compression quantified by Shannon entropy

Classical bit string

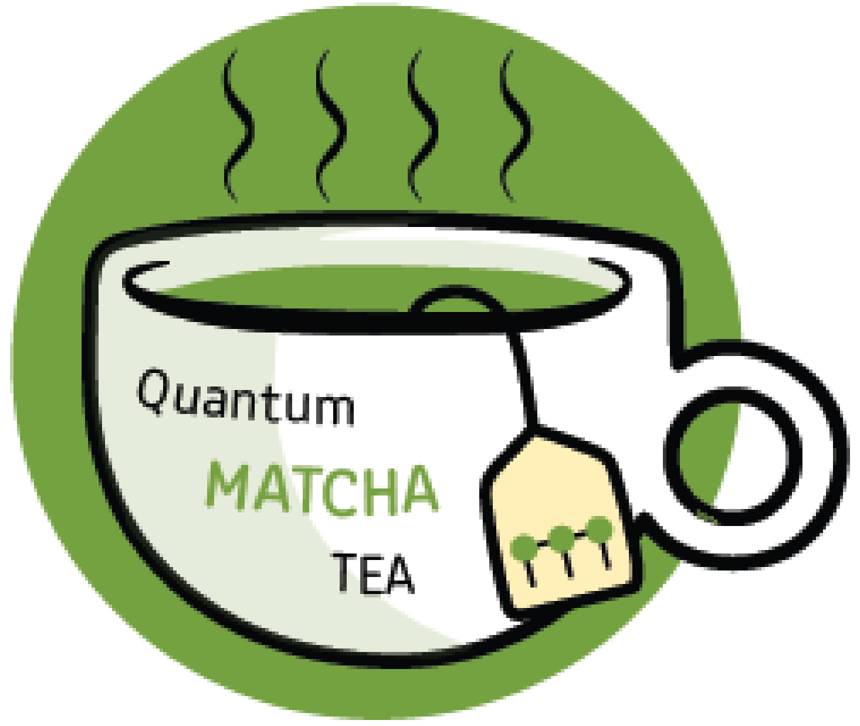
0000	0011	RANDOM
Optimal Compression	Here we can compress something. The quantum state is **entangled**, it cannot be written as classical state	No compression possible without approximations
$\vert 0000\rangle$	$\frac{1}{\sqrt{2}}(\vert 0000\rangle + \vert 1111\rangle)$	$\vert \text{RANDOM}\rangle$

Quantum state

Possible compression quantified by Von Neumann **entanglement** entropy

Image compression through SVD

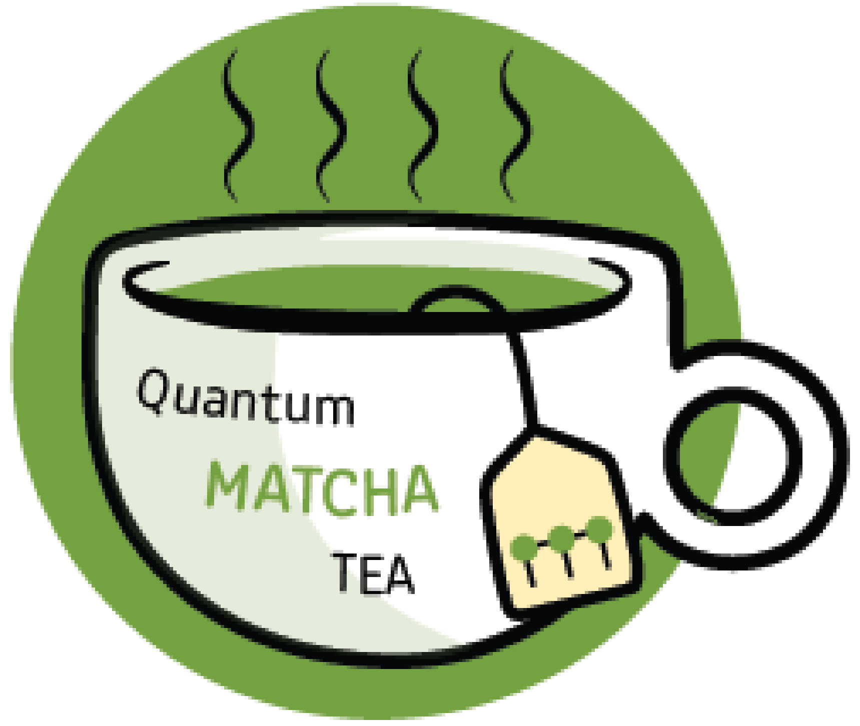
212 × 247
image



52364
pixels

Image compression through SVD

212 × 247
image



52364
pixels

$$\rightarrow \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0m} \\ a_{10} & a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \cdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nm} \end{pmatrix} \rightarrow U\,S\,V^T$$

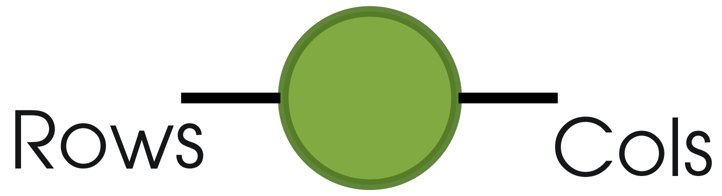
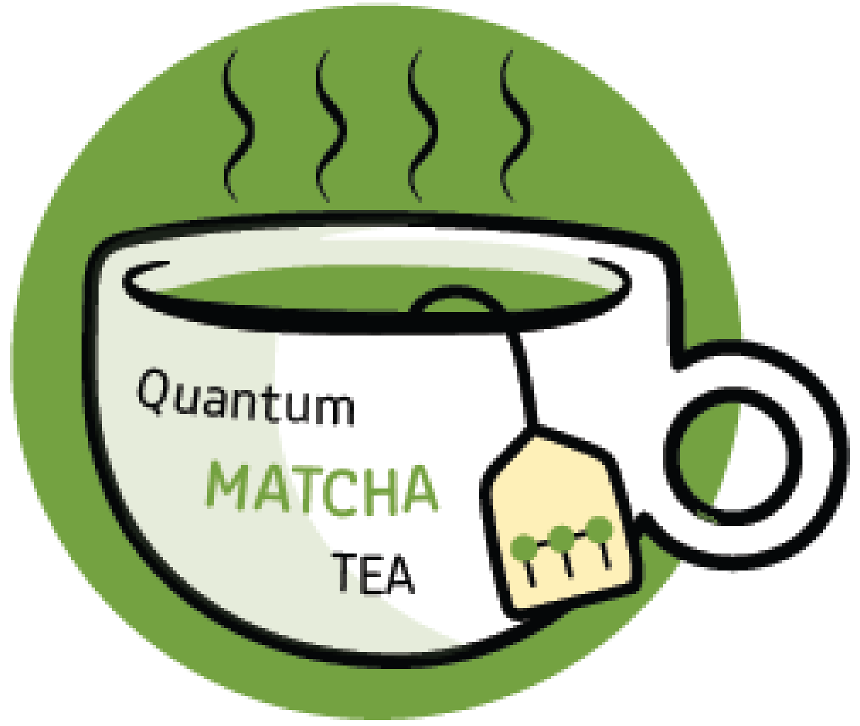


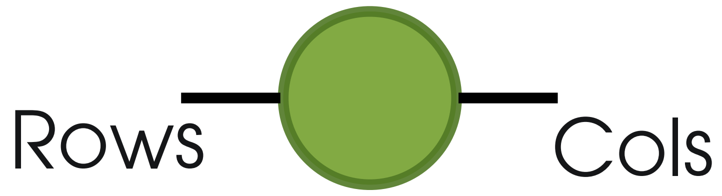
Image compression through SVD

212 × 247
image



52364
pixels

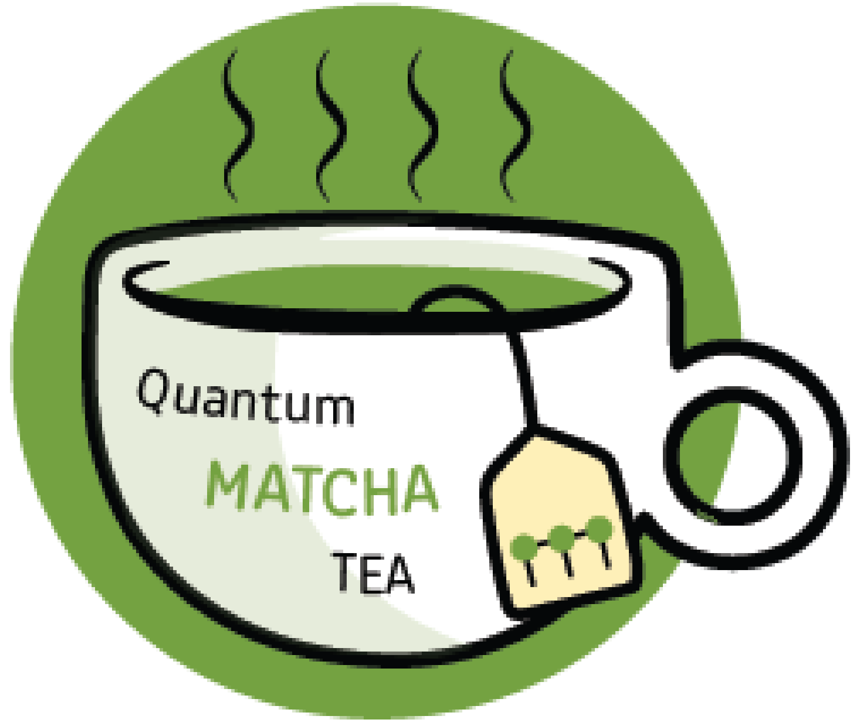
$$\rightarrow \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0m} \\ a_{10} & a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \cdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nm} \end{pmatrix} \rightarrow U\,S\,V^T$$



$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

Image compression through SVD

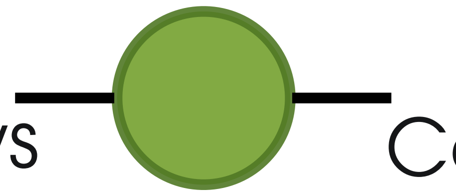
212 × 247
image



52364
pixels

$$\begin{pmatrix} a_{00} & a_{01} & --- & a_{0m} \\ a_{10} & a_{11} & --- & a_{1m} \\ \vdots & \ddots & --- & \vdots \\ a_{n0} & a_{n1} & --- & a_{nm} \end{pmatrix} \rightarrow U\,S\,V^T$$

Singular value
decomposition

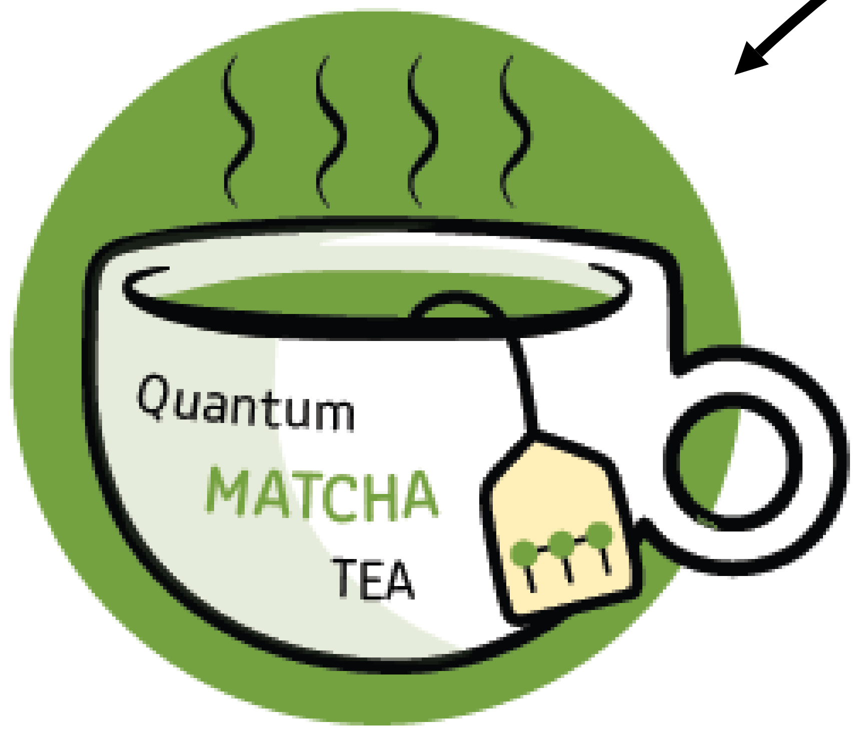


$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

Image compression through SVD

No truncation

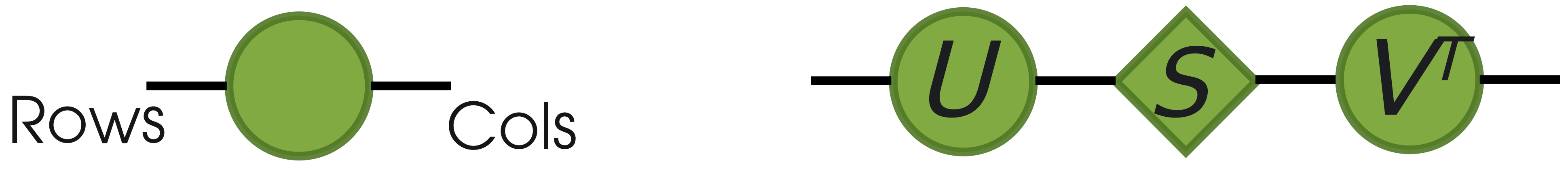
212 × 247 image



52364 pixels

$$\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0m} \\ a_{10} & a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \cdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nm} \end{pmatrix} \rightarrow USV^T$$

Singular value decomposition



$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

Image compression through SVD

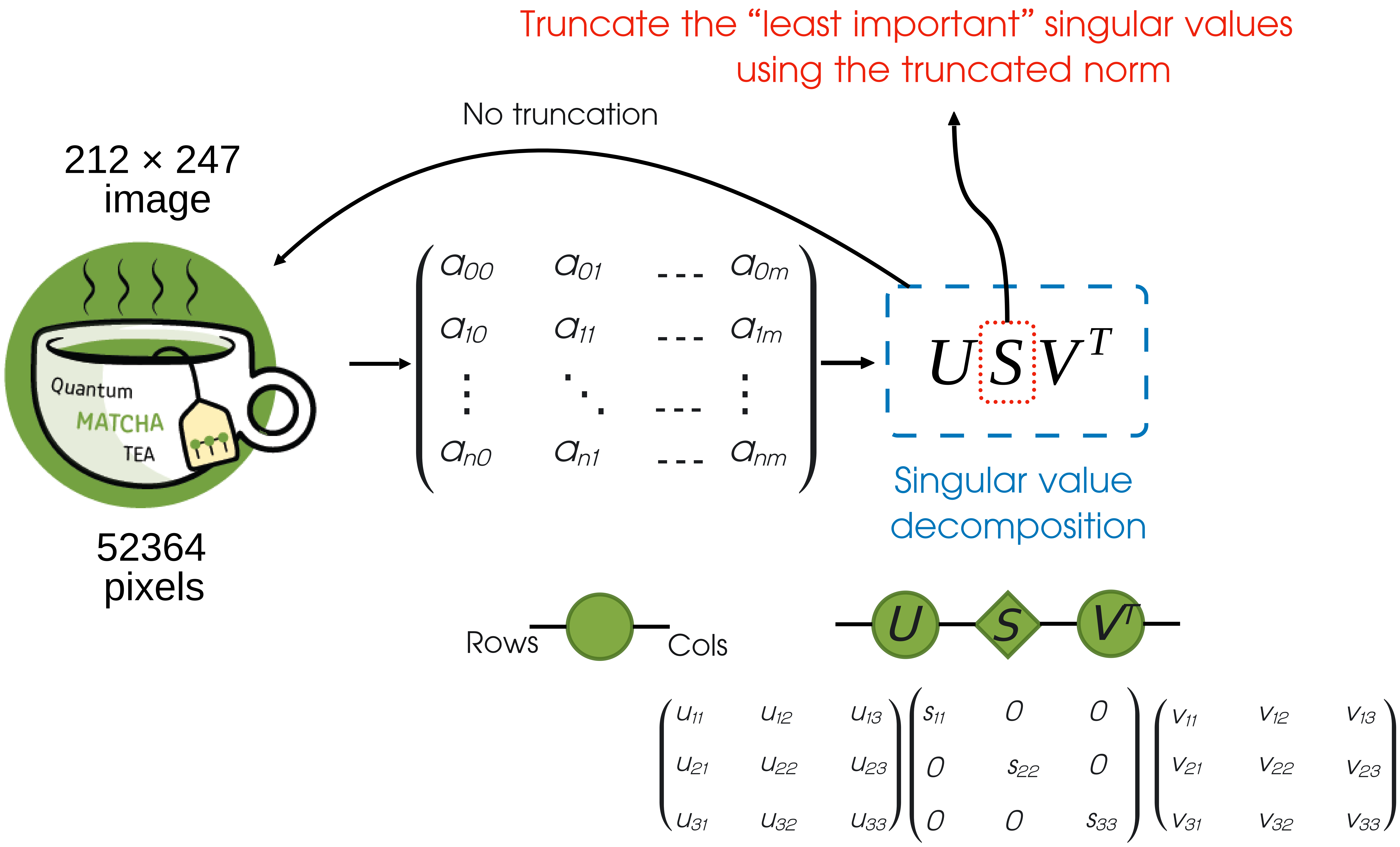
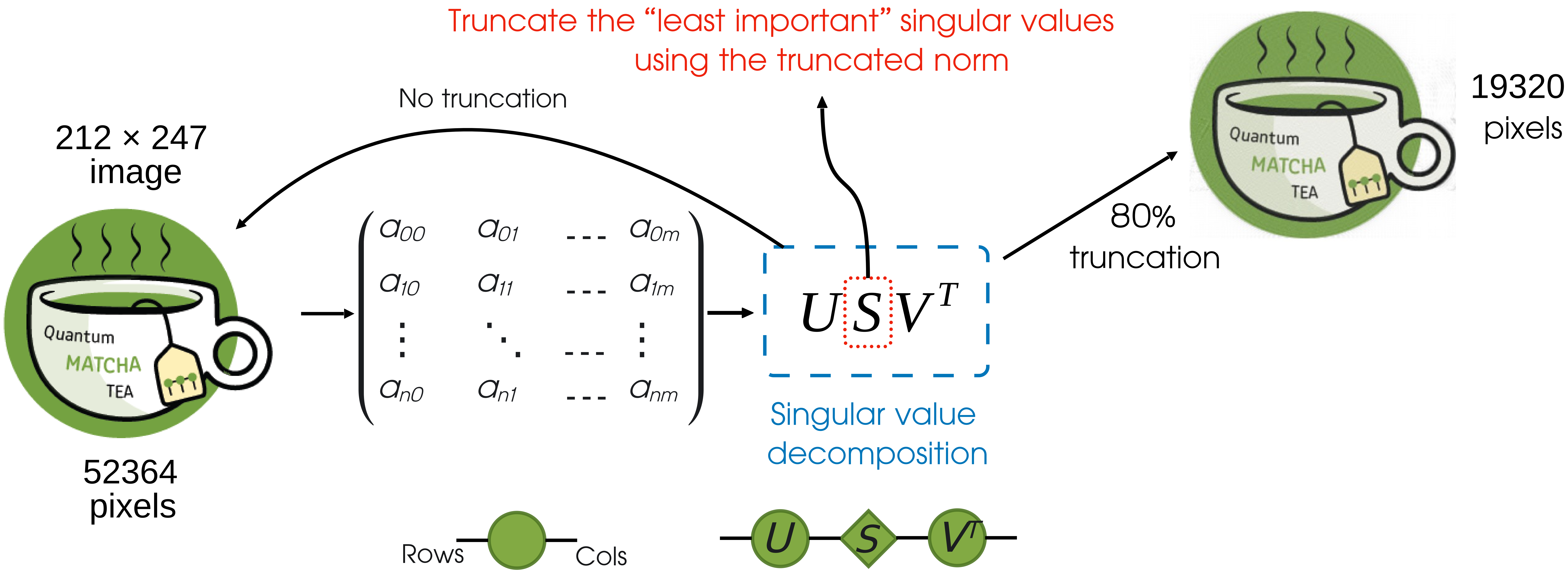


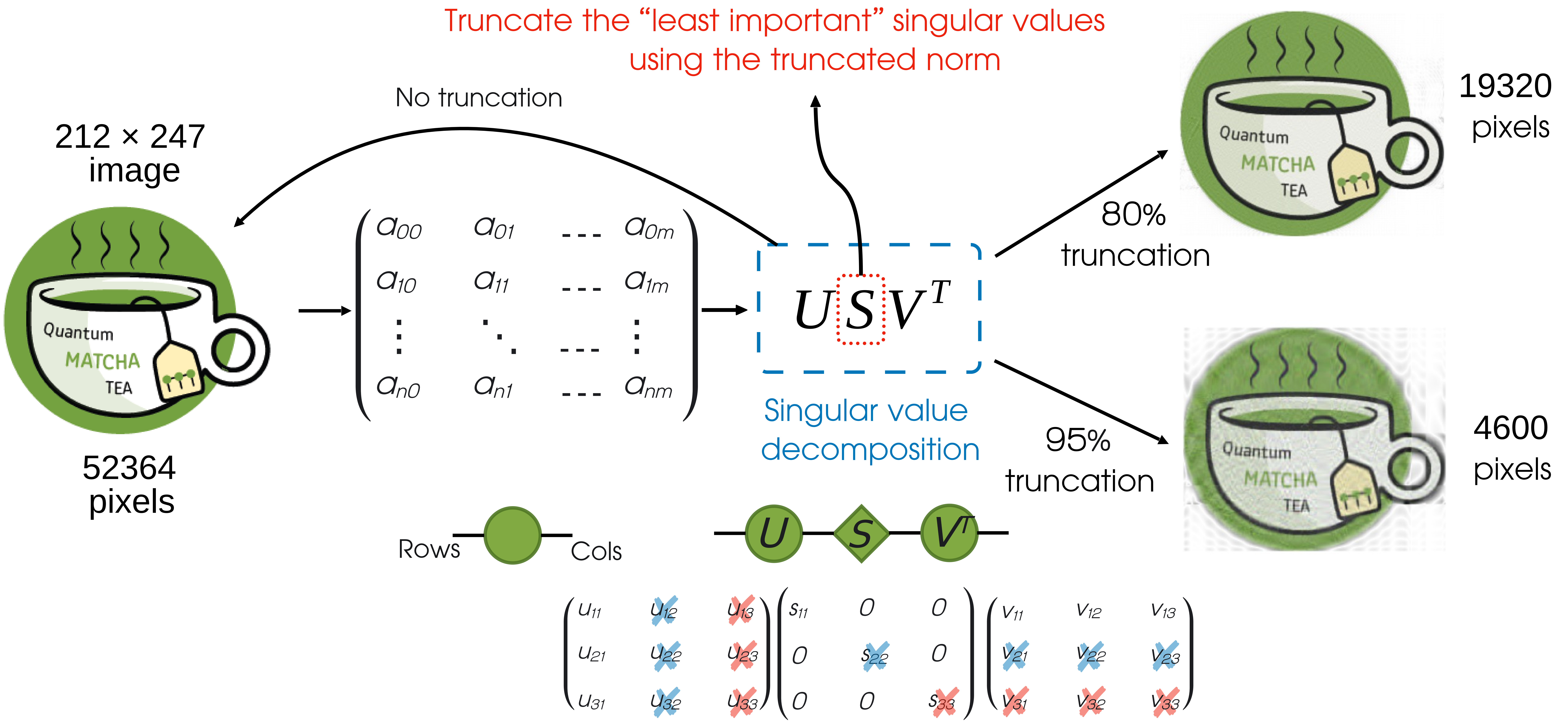
Image compression through SVD



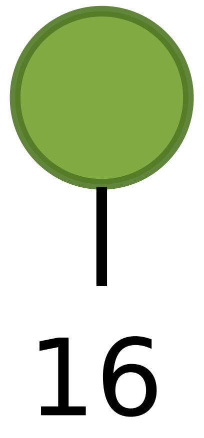
$$\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0m} \\ a_{10} & a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \cdots & \vdots \\ a_{n0} & a_{n1} & \cdots & a_{nm} \end{pmatrix}$$

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ u_{21} & u_{22} & u_{23} \\ u_{31} & u_{32} & u_{33} \end{pmatrix} \begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix} \begin{pmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{pmatrix}$$

Image compression through SVD

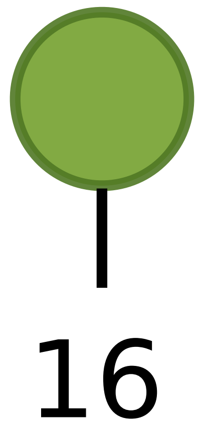


Compressing a 4-qubits quantum state

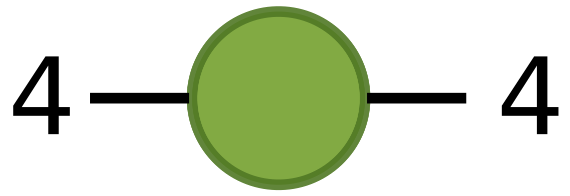


$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

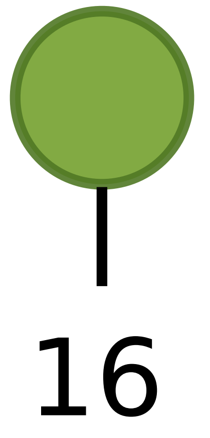
Compressing a 4-qubits quantum state



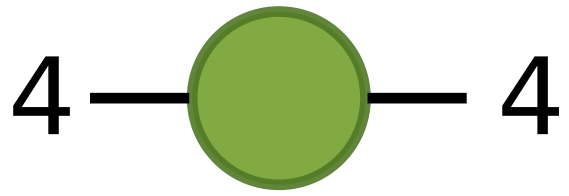
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\\vdots\\0\\1\end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$$

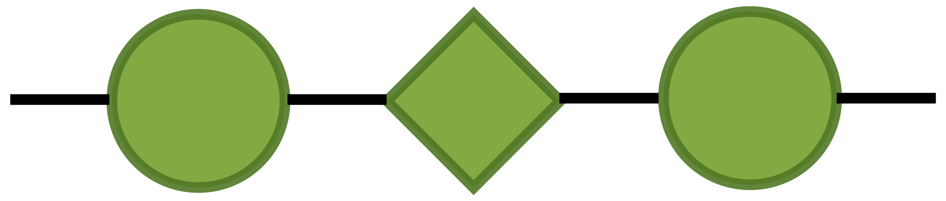


Compressing a 4-qubits quantum state



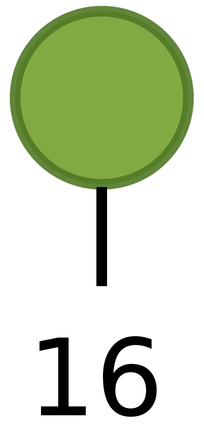
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\\vdots\\0\\1\end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$$



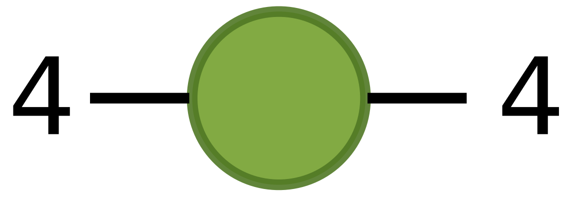


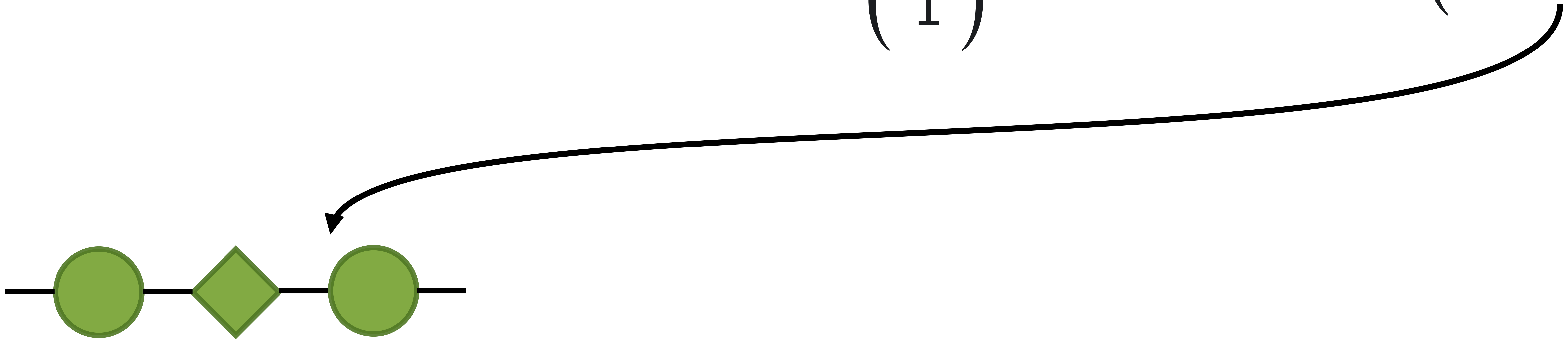
$$\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix}$$

Compressing a 4-qubits quantum state



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\\vdots\\0\\1\end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$$

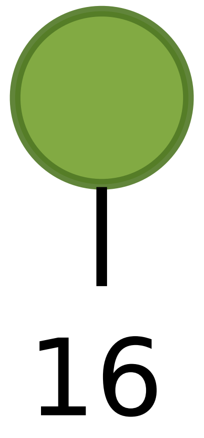




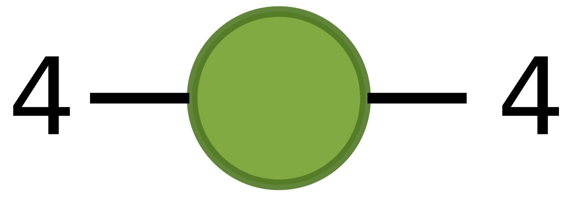
$$\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix}$$

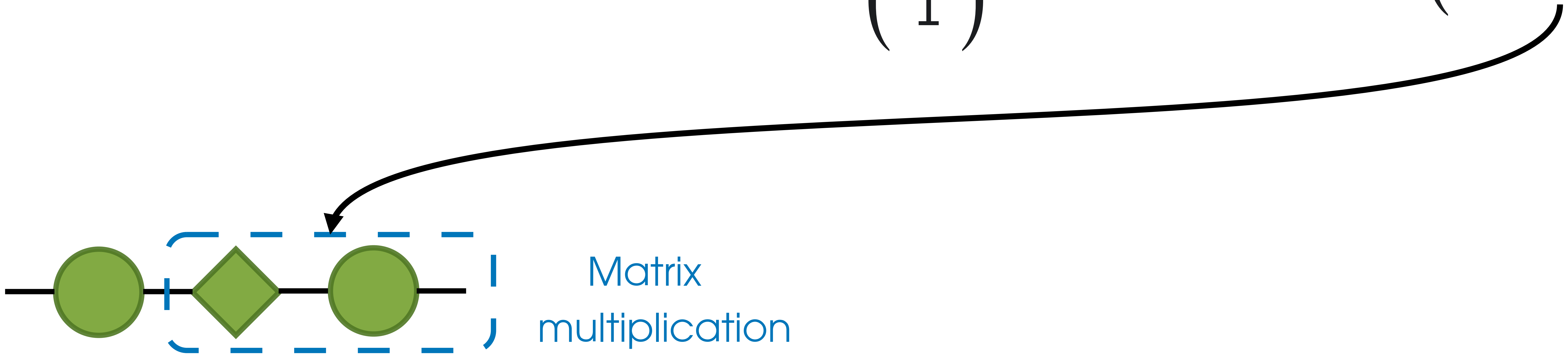
$4 \rightarrow 2$

Compressing a 4-qubits quantum state



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\\vdots\\0\\1\end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}$$

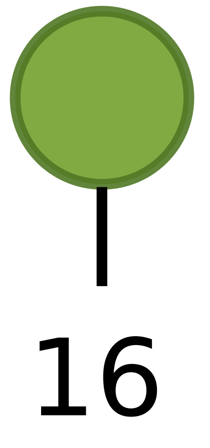




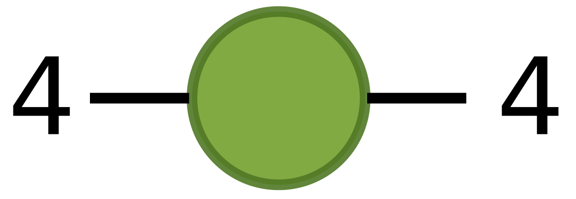
$$\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&0&0&1\\0&1&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix}$$

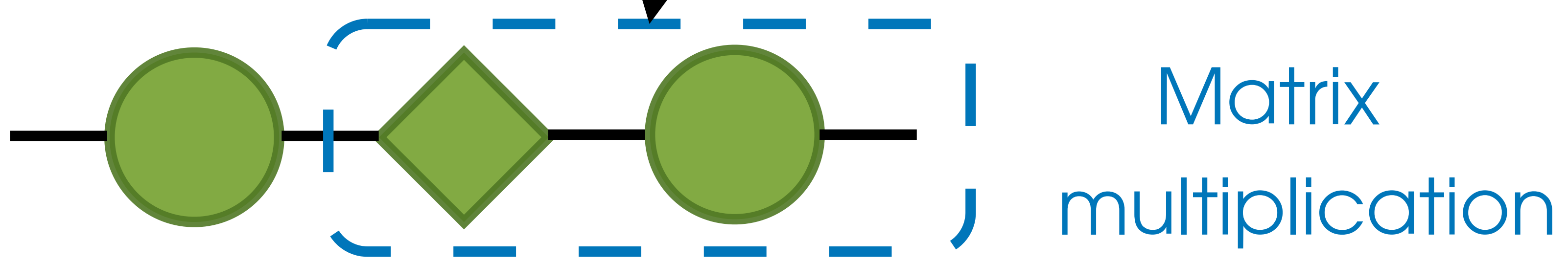
$4 \rightarrow 2$

Compressing a 4-qubits quantum state



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

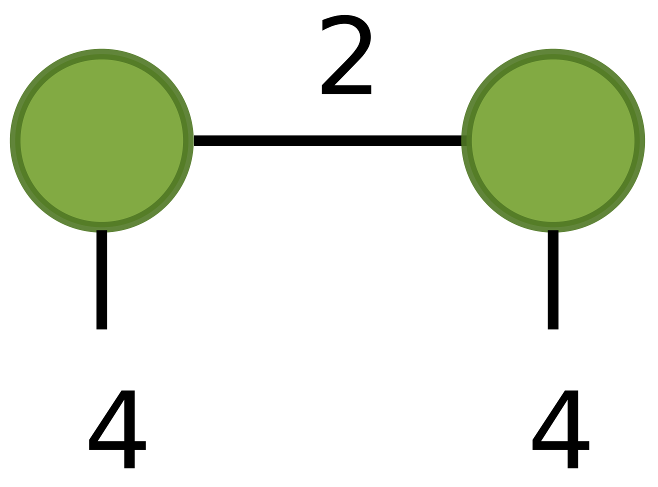




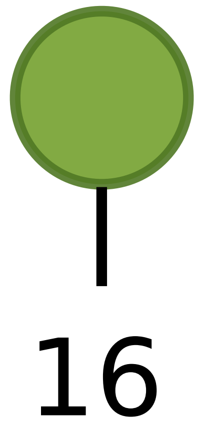
$$\begin{pmatrix} 1 & 0 & \cancel{0} & \cancel{0} \\ 0 & 0 & \cancel{1} & \cancel{0} \\ 0 & 0 & \cancel{0} & \cancel{1} \\ 0 & 1 & \cancel{0} & \cancel{0} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cancel{0} & 0 \\ 0 & 0 & 0 & \cancel{0} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} \end{pmatrix}$$

$4 \rightarrow 2$

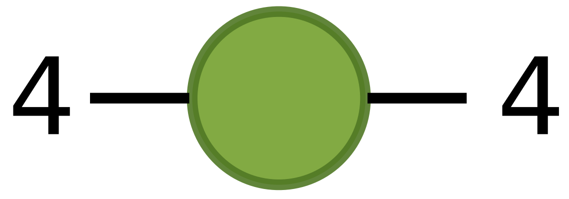
$\longrightarrow$

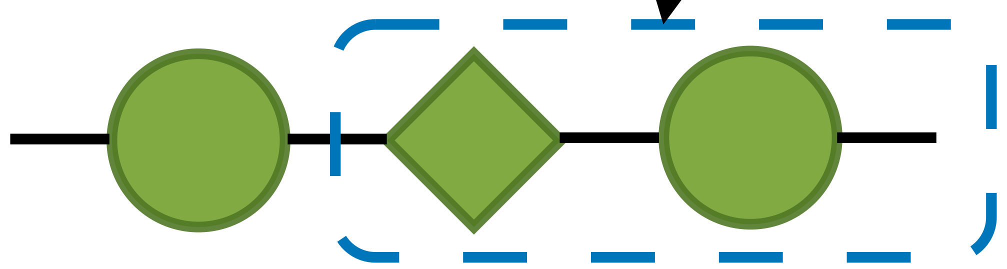


Compressing a 4-qubits quantum state



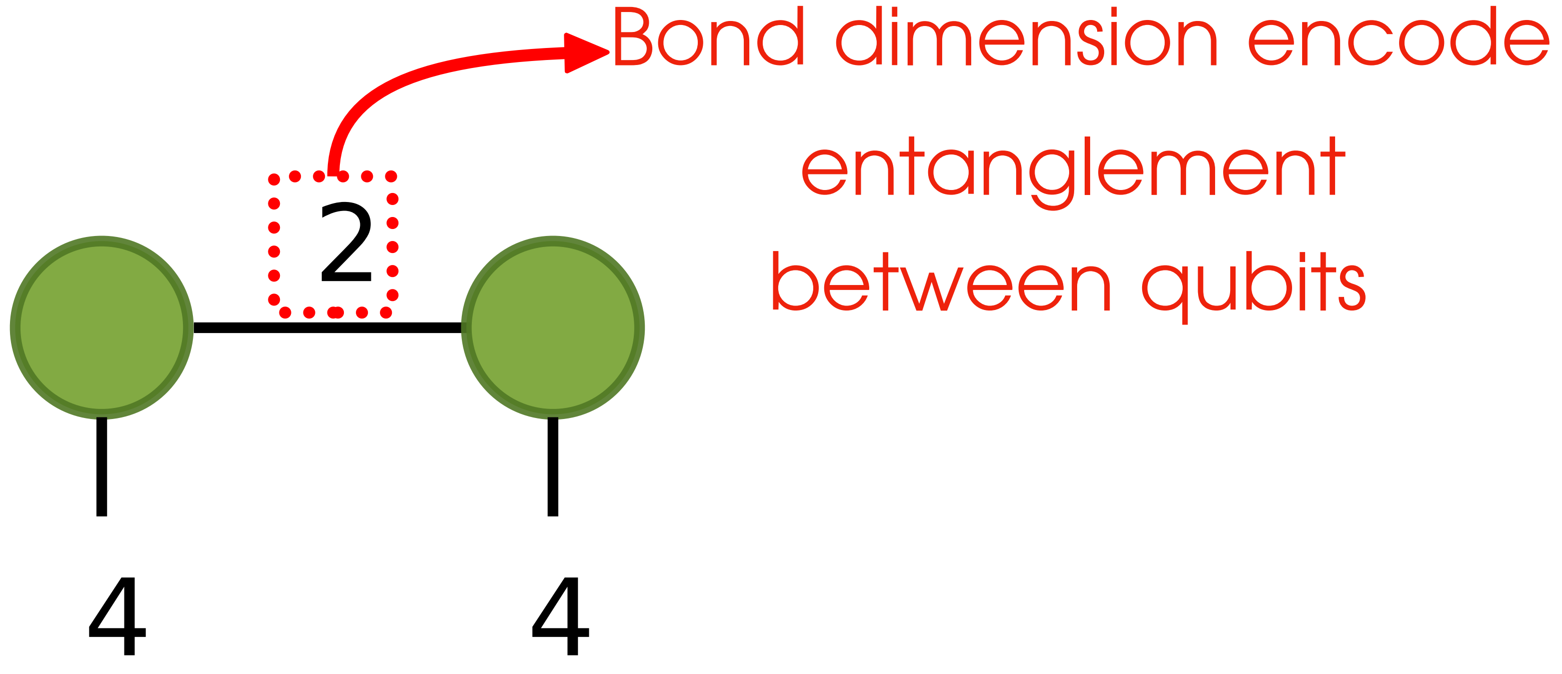
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \longrightarrow \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



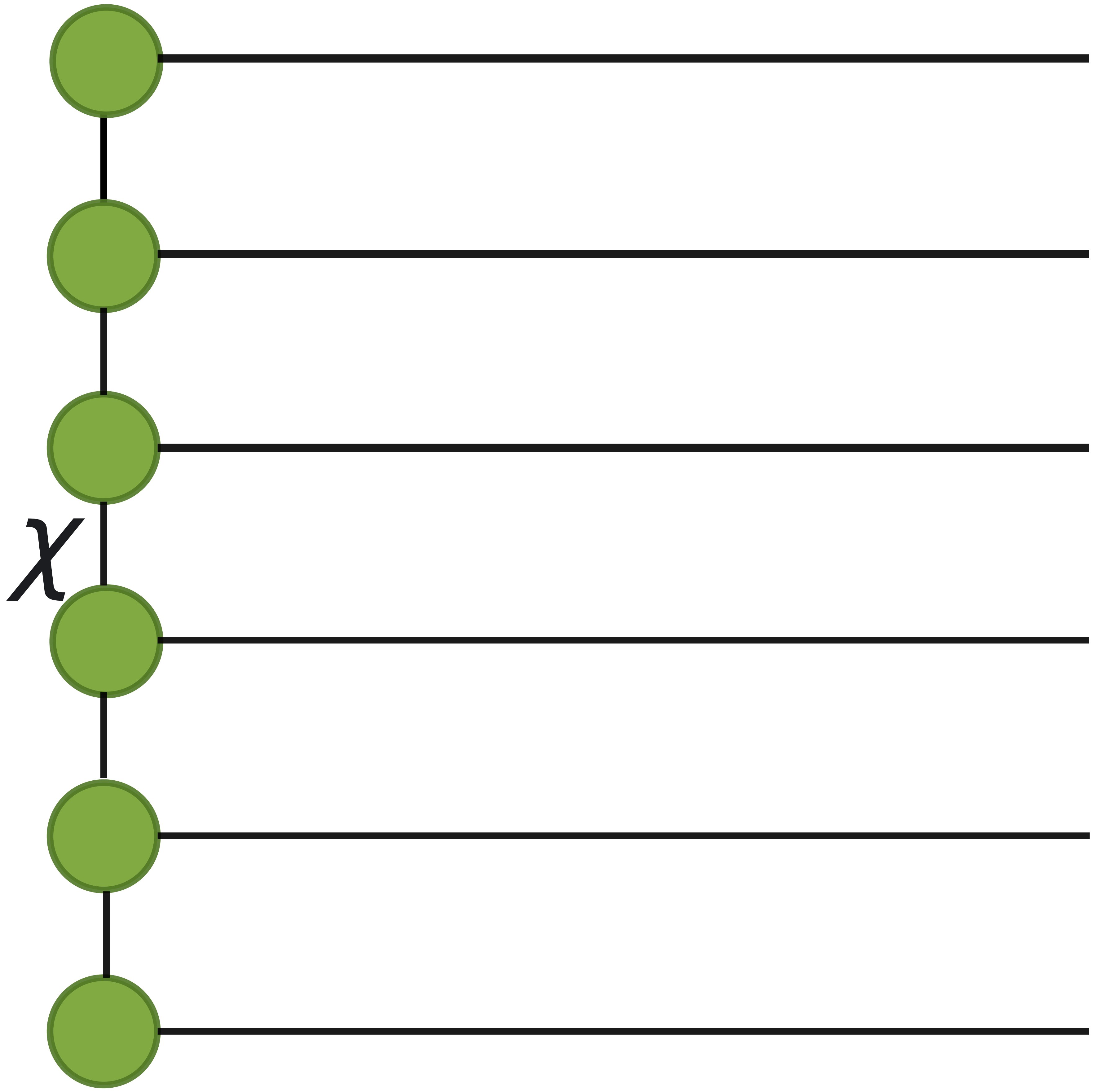


$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

4 → 2



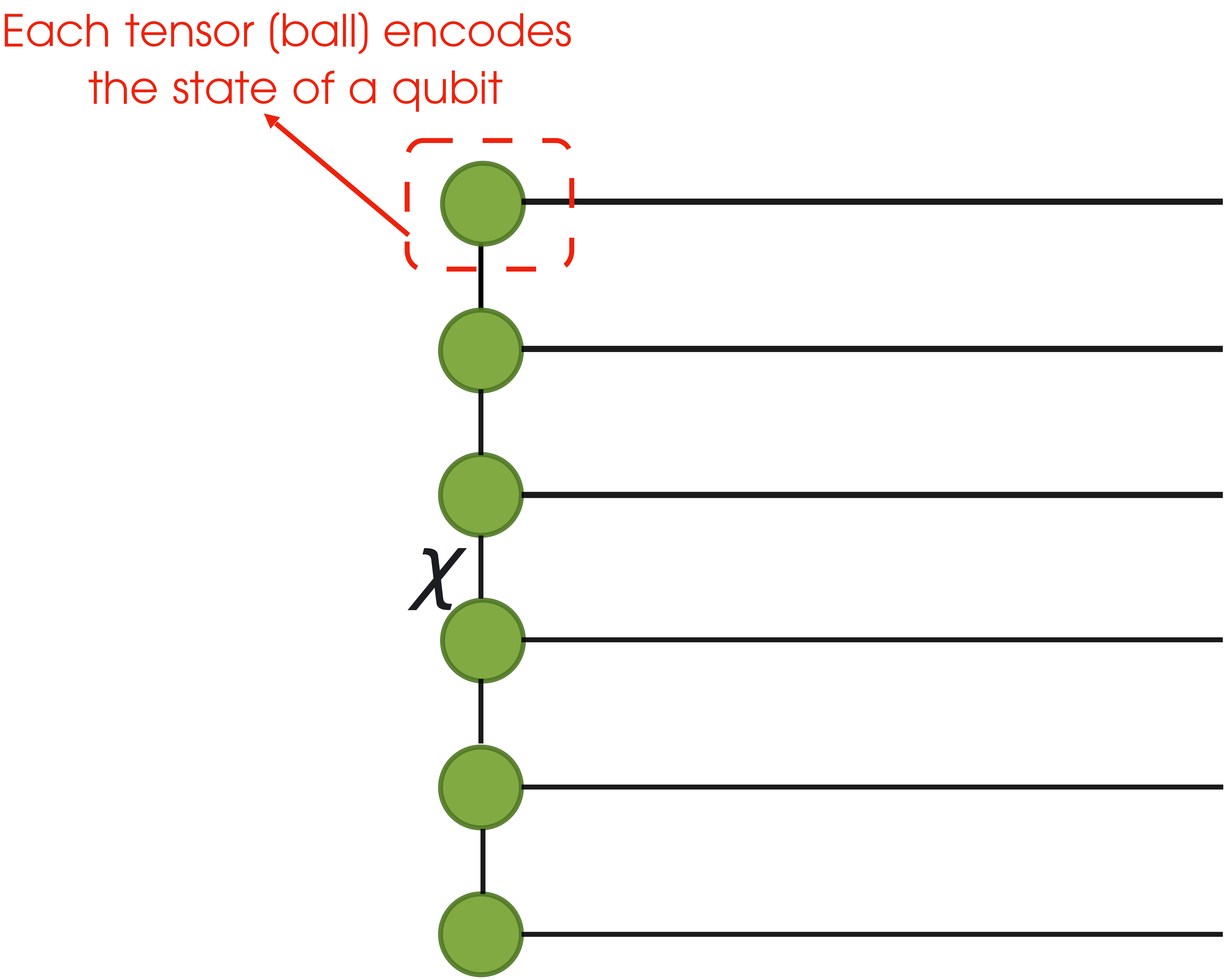
Matrix Product States



Memory requirements

$$O(2^n) \rightarrow O(2n\,\chi^2)$$

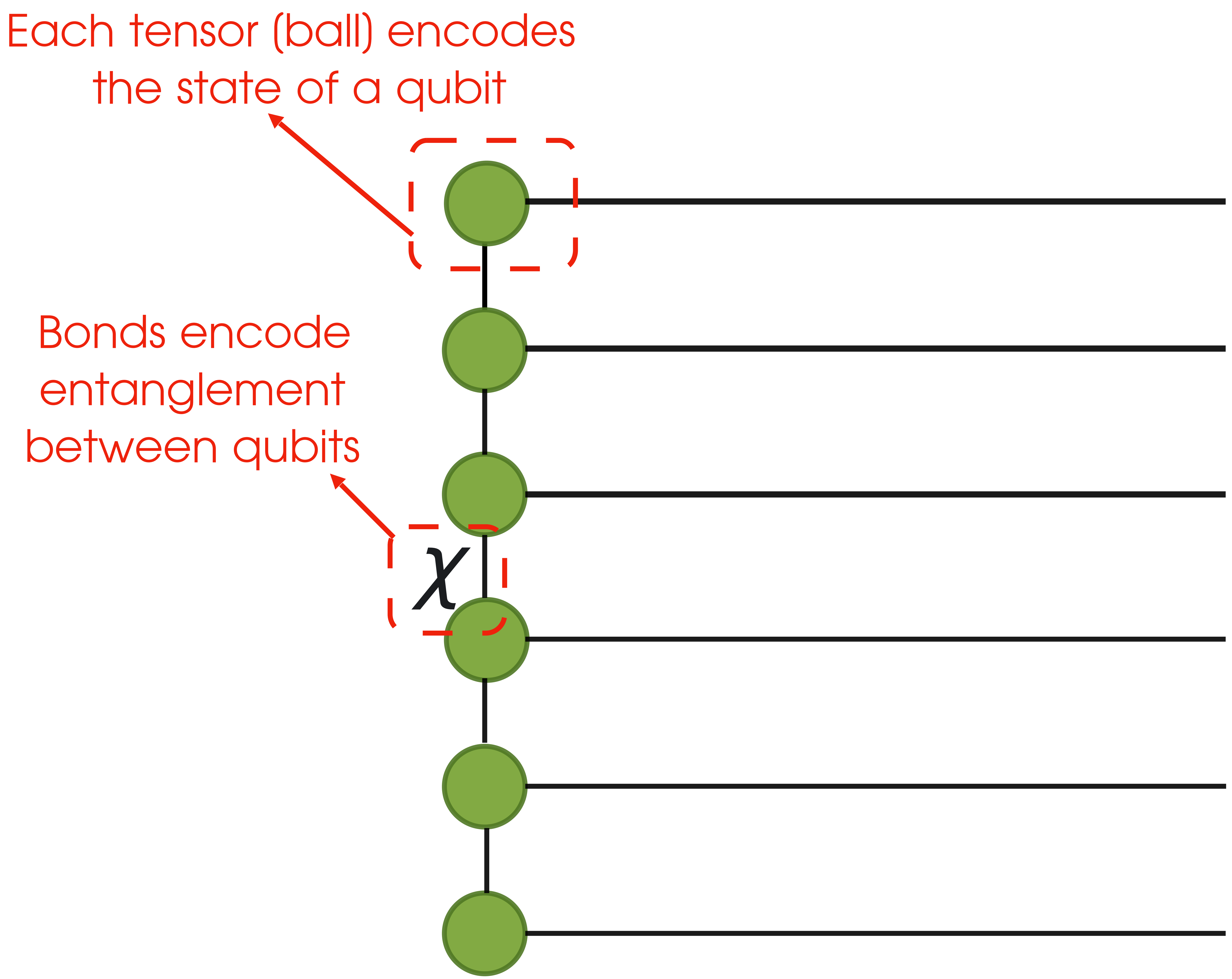
Matrix Product States



Memory requirements

$$O\left(2^n\right) \rightarrow O\left(2n\,\chi^2\right)$$

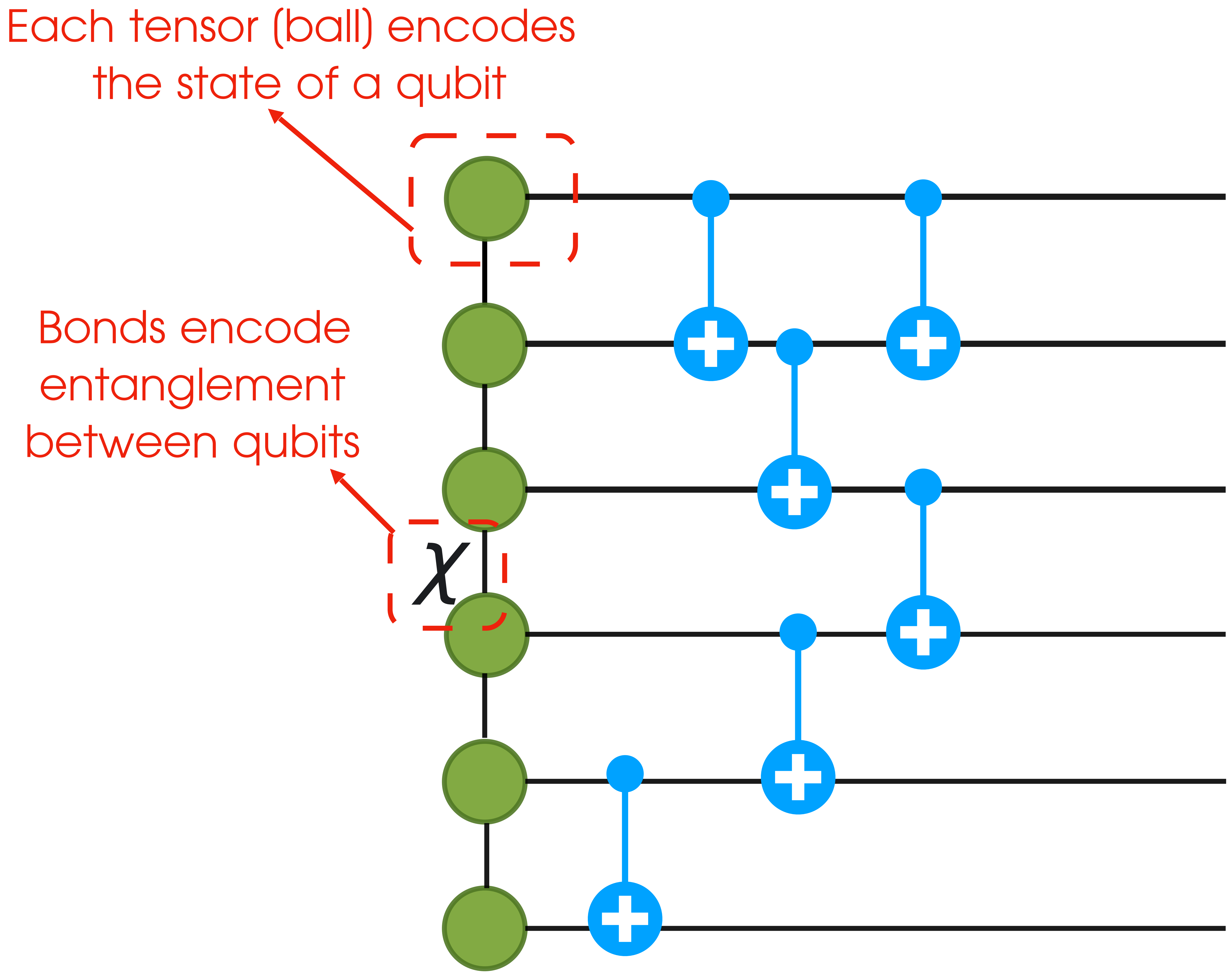
Matrix Product States



Memory requirements

$$O\left(2^n\right) \rightarrow O\left(2n\,\chi^2\right)$$

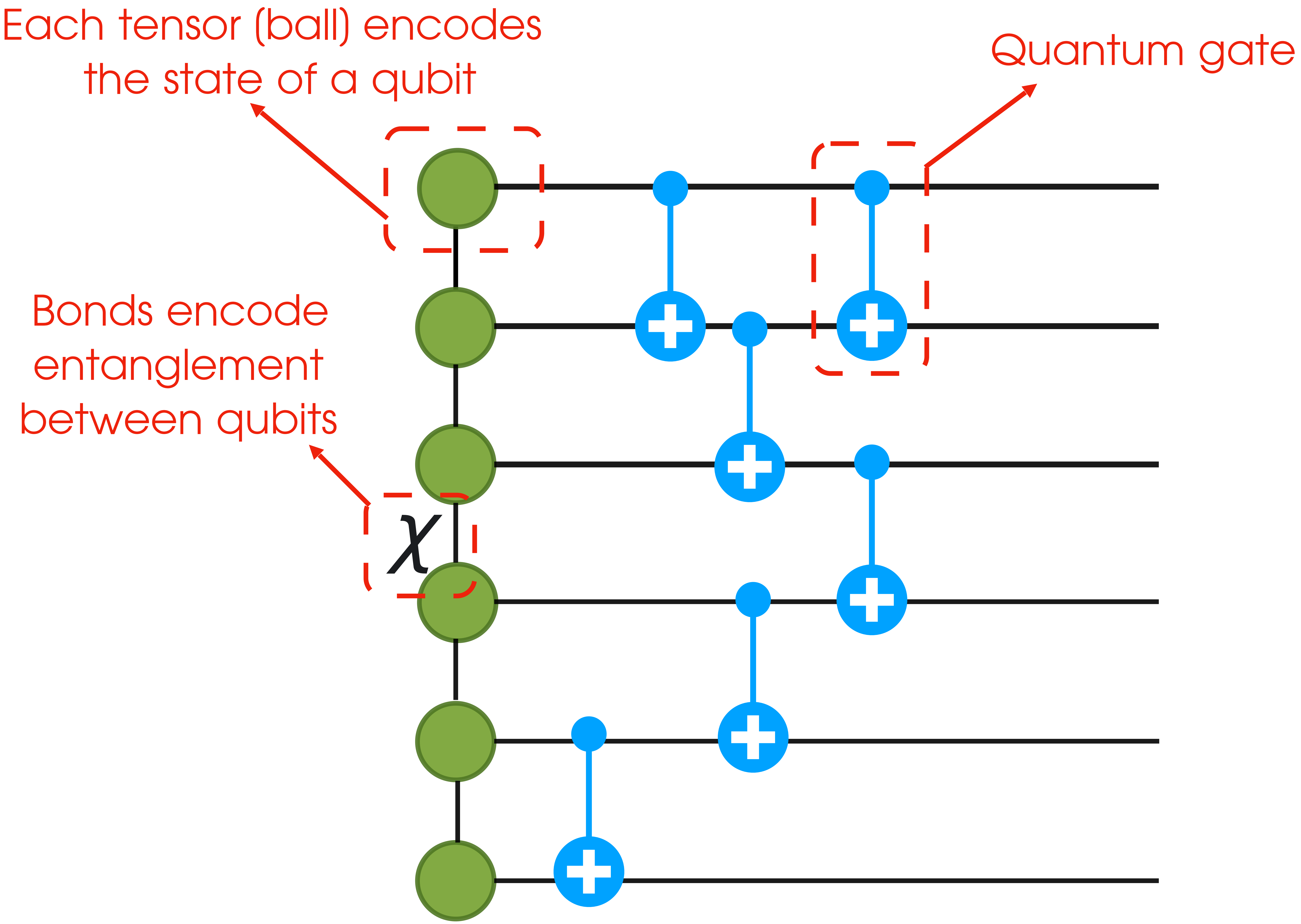
Matrix Product States



Memory requirements

$$O(2^n) \rightarrow O(2n\,\chi^2)$$

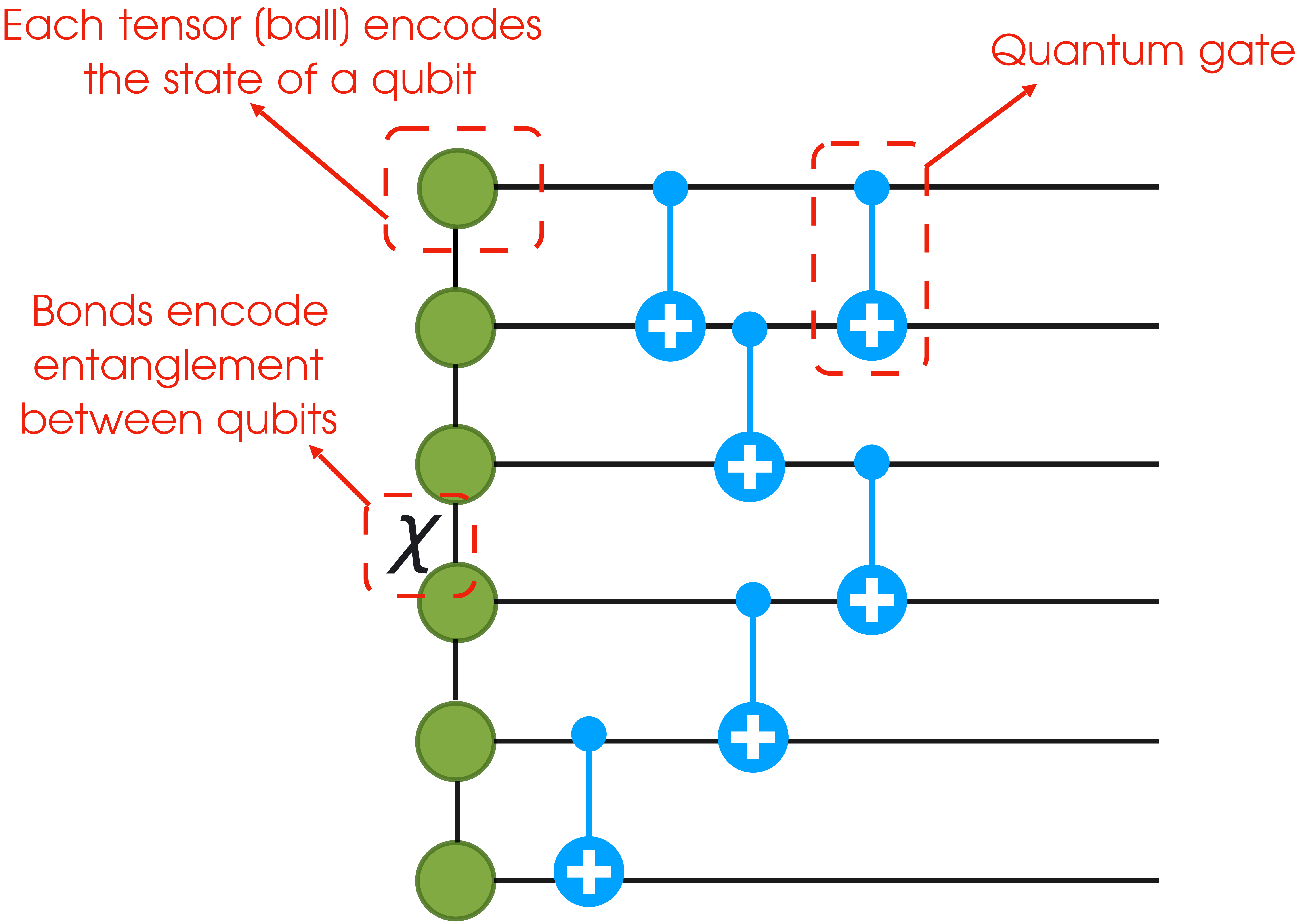
Matrix Product States



Memory requirements

$$O\left(2^n\right) \rightarrow O\left(2n\,\chi^2\right)$$

Matrix Product States



Memory requirements

$$O\left(2^n\right) \rightarrow O\left(2n\,\chi^2\right)$$

MPS SIMULATIONS ARE NOT LIMITED BY THE NUMBER OF QUBITS BUT BY THE ENTANGLEMENT

QuantumTEA distribution

Tensor network ← **T** **E** **A** → Applications

E ↓ Emulator

Quantum tea leaves: **Utility**

Quantum matcha tea: **quantum circuit HPC simulations**

Quantum red tea: **tensor handling**

Quantum chai tea: **AI and ML with tensor networks**

Quantum green tea: **Schrödinger equation solution for many-body states**

QuantumTEA distribution

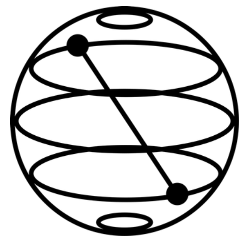
T → Tensor network
E → Emulator
A → Applications

Public!

- Quantum tea leaves: **Utility**
- Quantum matcha tea: **quantum circuit HPC simulations**
- Quantum red tea: **tensor handling**
- Quantum chai tea: **AI and ML with tensor networks**
- Quantum green tea: **Schrödinger equation solution for many-body states**

Quantum Matcha TEA workflow

Quantum circuit

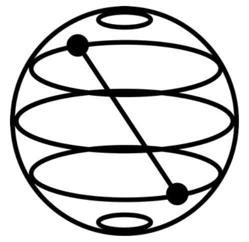


Observables

Python interface, definition of the problem

Quantum Matcha TEA workflow

Quantum circuit



Observables

→

Matrix product state simulator

Python interface, definition of the problem

Quantum Matcha TEA workflow

Quantum circuit

Observables

Python interface, definition of the problem

Matrix product state simulator

Backends for running the simulations

NumPy

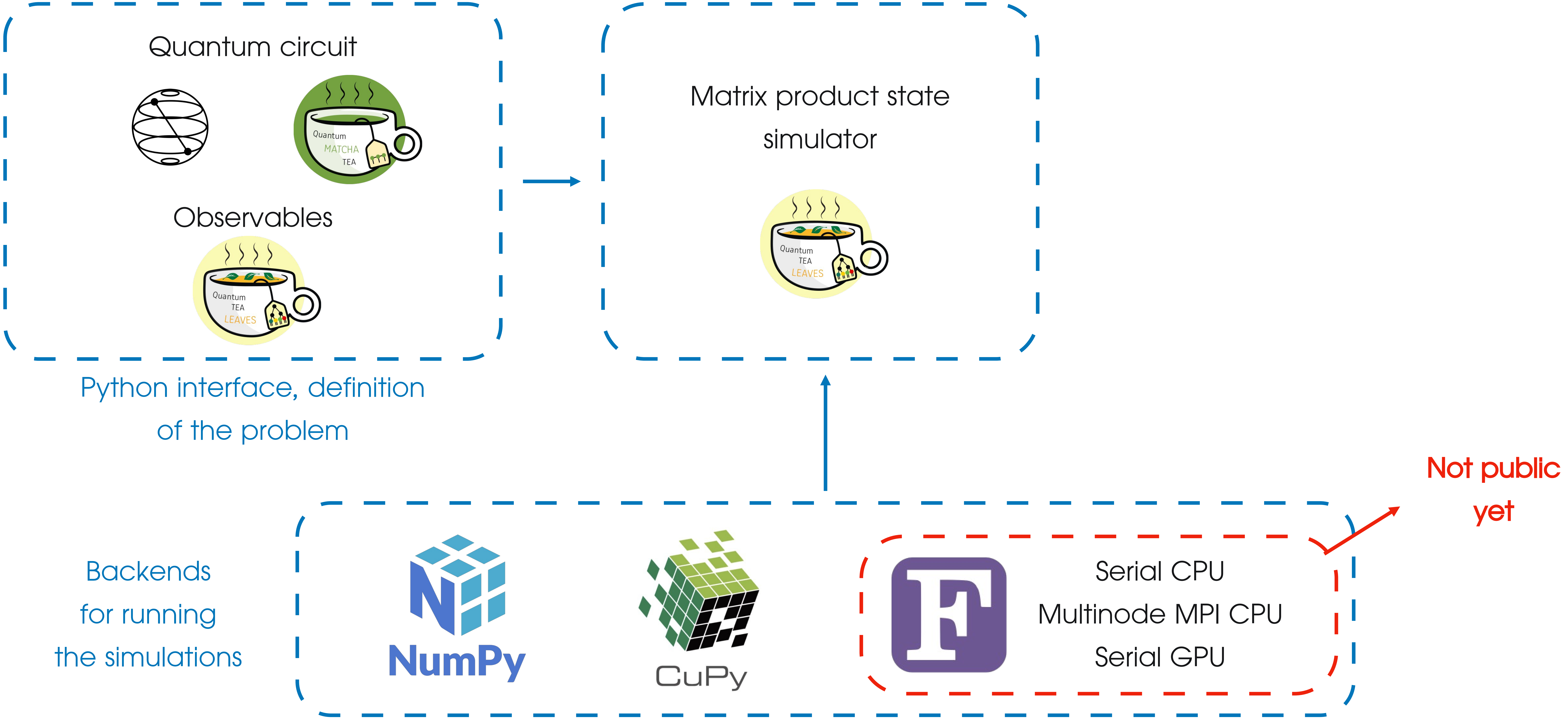
CuPy

Serial CPU

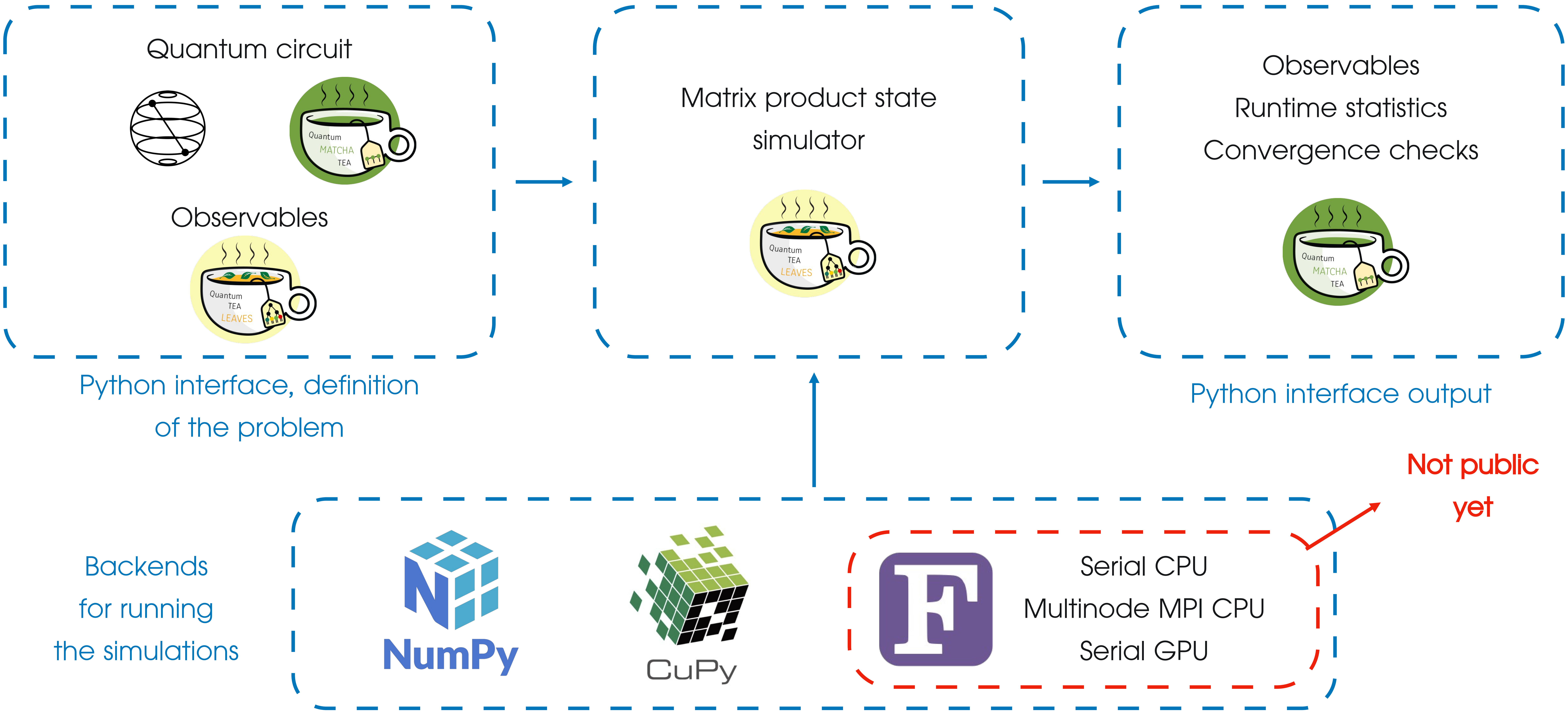
Multinode MPI CPU

Serial GPU

Quantum Matcha TEA workflow

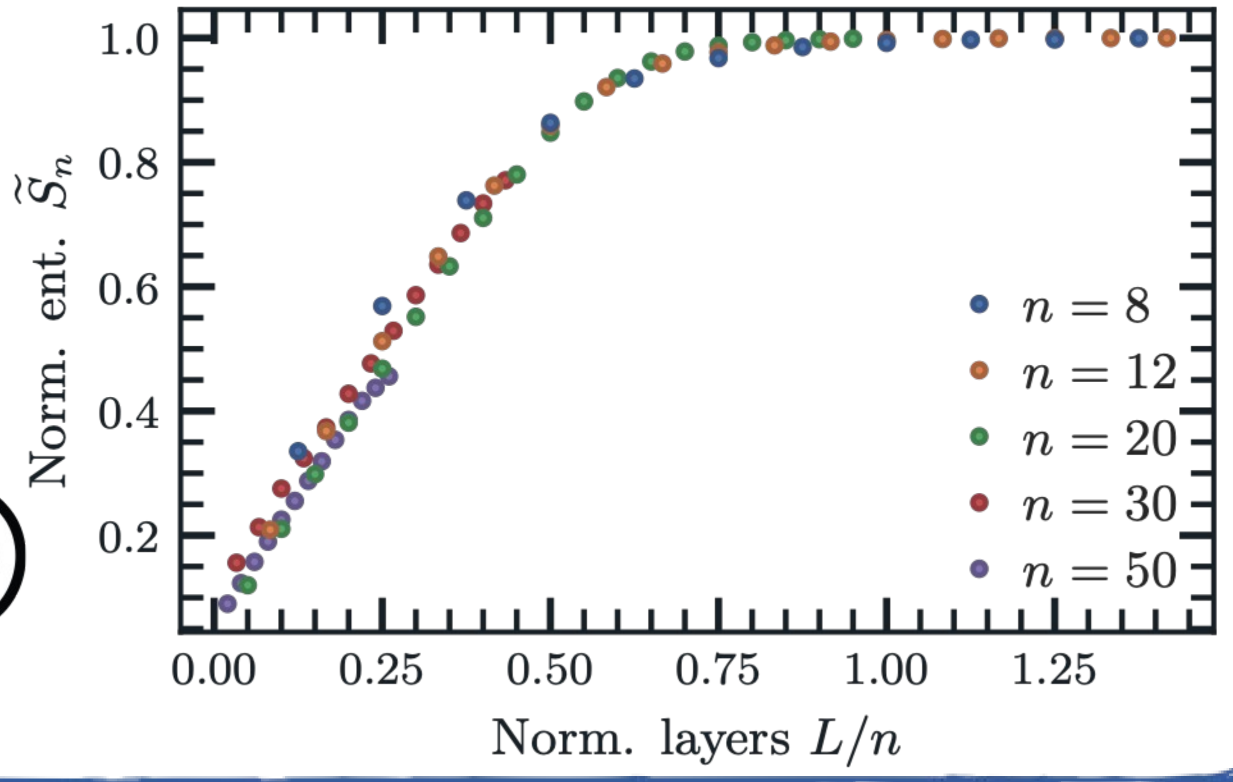


Quantum Matcha TEA workflow

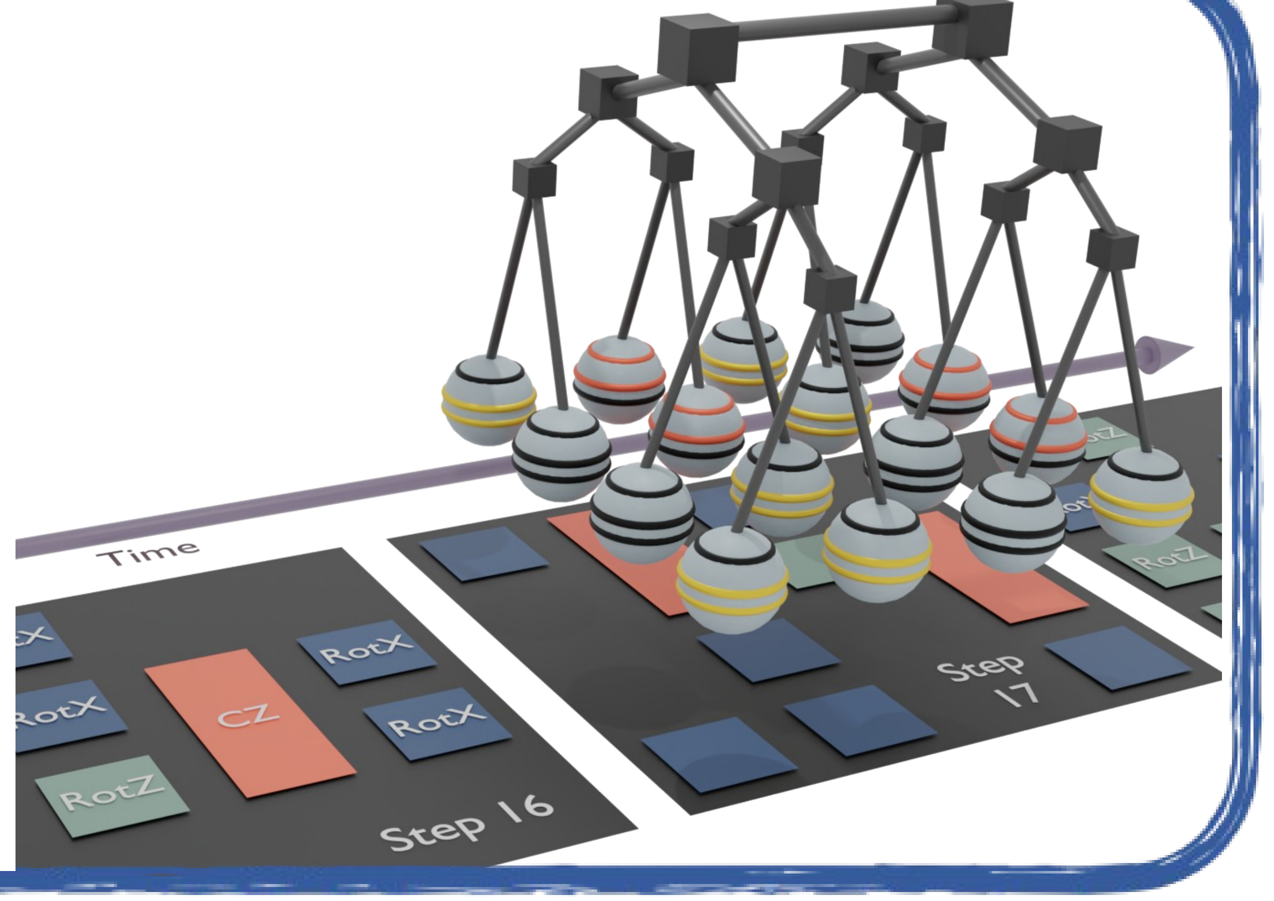


Applications

Entanglement entropy production in QNN
Ballarin, Marco, et al. arXiv:2206.02474
- Simulations up to 50 qubits
- Bond dimension of 4096
- 11h of runtime on Galileo100



Ab initio two-dimensional digital twin for quantum computer
Jaschke, Daniel, et al. arXiv:2210.03763
- Use of the unbiased sampling
- Quantum matcha tea simulations used as target state to compute the fidelity of a simulation with crosstalk



QuantumTEA Cloud Platform

- Prototype that provides the QuantumTEA's capabilities as Cloud Service
  - QuantumTEA-as-a-Service
  - based on Kubernetes running at CloudVeneto
- It is meant to be easily accessible
- It allows users to run quantum circuits with QuantumTEA without specific cloud computing skills

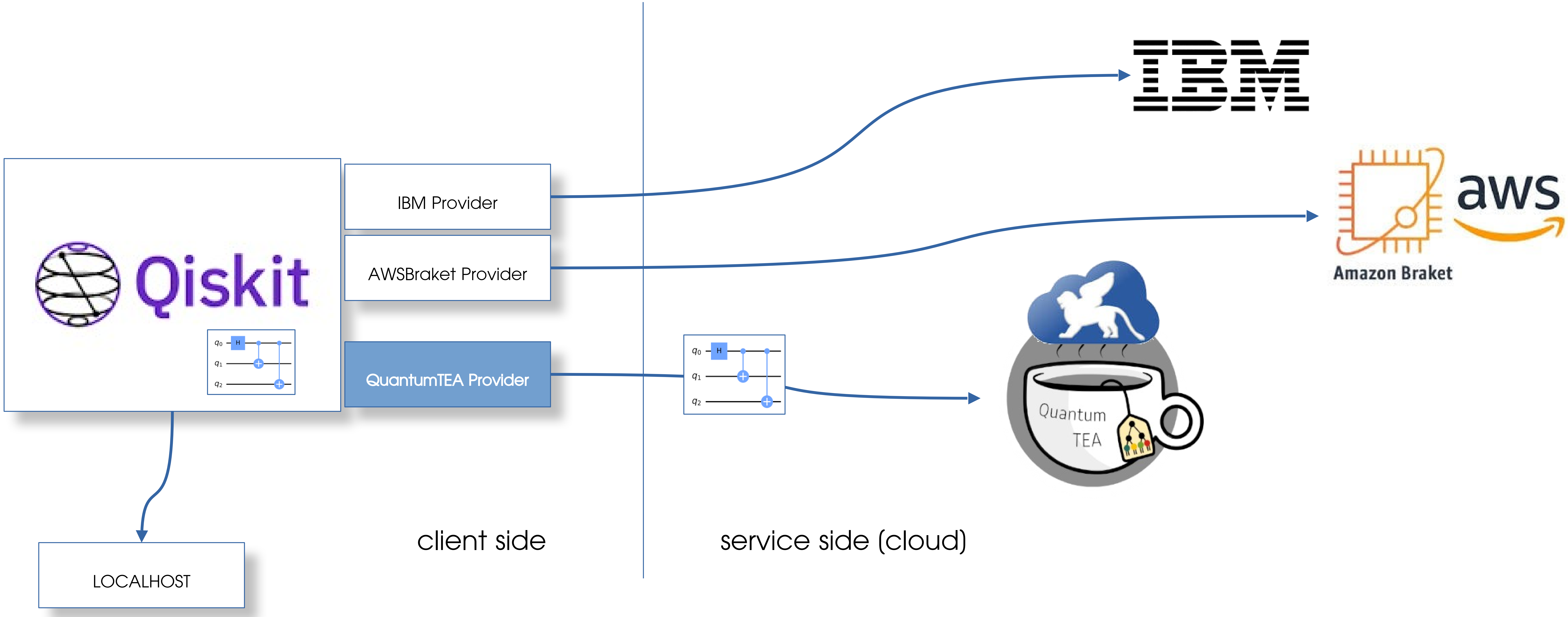
Quantum computing standards and Qiskit

- No quantum computing standards exists
  - several technologies, APIs, provider specific

- **Qiskit** open source Python SDK developed by IBM
  - Qiskit provides tools for creating and manipulating quantum programs
  - and running them on
    - real quantum computers
    - simulators on a local computer or on a remote cloud service

Qiskit high level architecture

Users can take their existing algorithms written in Qiskit and, with a few lines of code, run them directly on different cloud platform including our QuantumTEA



QuantumTEA Qiskit provider implementation

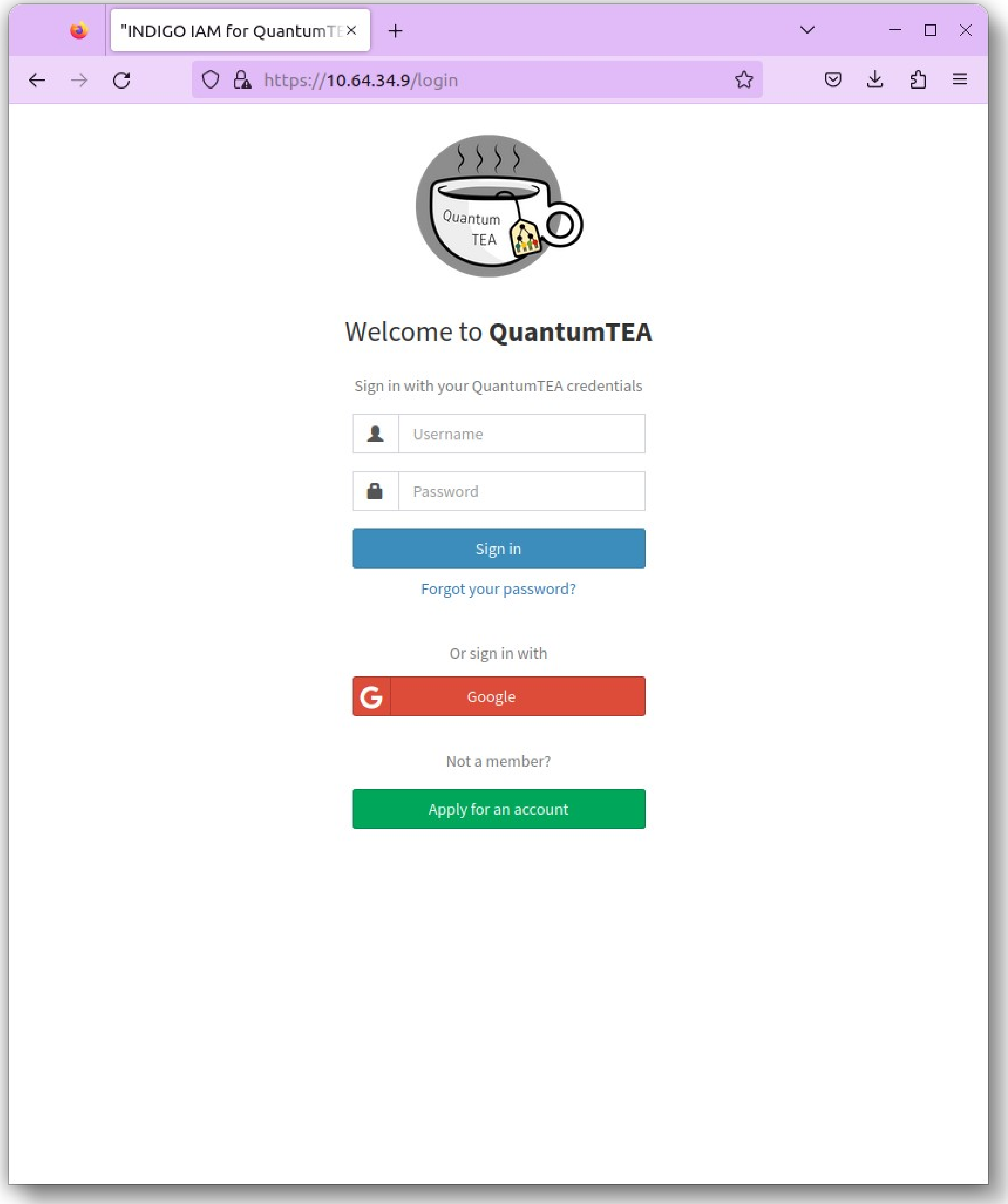
- QuantumTEA-qiskit-provider implements a set of Qiskit interfaces (qiskit.providers Python APIs)
  - **Provider** subclass that handles access to the backend(s)
    - handles backend objects that enable executing circuits on a device or simulator (initialization, authentication, etc)
  - **Backend** subclass and its run() method
    - provide the interface between Qiskit and the hardware or simulator that will execute circuits
    - run() method handles submitting the circuits to the backend to be executed and returning a Job objec (involves serialization (JSON), low level communication layer (REST))
  - **Job** subclass that handles interacting with a running job
    - the output from the run() method

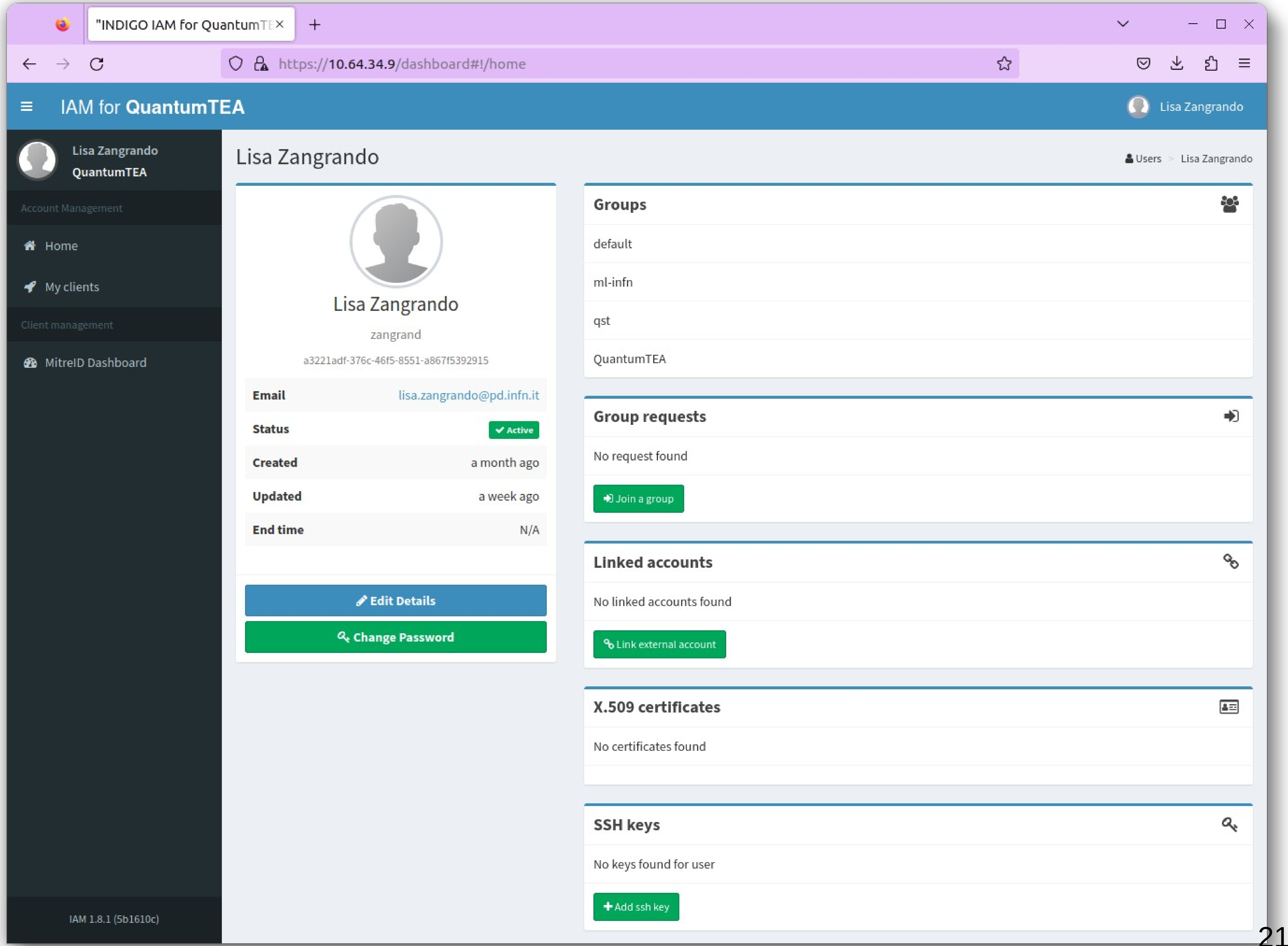
QuantumTEA Qiskit provider implementation

```
from qiskit import QuantumCircuit
from qt_provider import QuantumTeaProvider
# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)
# Add a H gate on qubit 0 and a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.h(0)
circuit.cx(0, 1)
# Map the quantum measurement to the classical bits
circuit.measure([0, 1], [0, 1])
# Create a new QuantumTeaProvider instance
qt_provider = QuantumTeaProvider(TOKEN)
# Get the “KubernetesBackend” backend
qt_backend = qt_provider.get_backend(name="QuantumMatchaTEA")
# Execute the circuit on the QuantumTea Cloud Platform
job = qt_backend.run(qc)
# Grab results from the job
result = job.result()
```

To access the QT service you need an ACCESS TOKEN (IAM or Keystone)

Access Token from IAM





QuantumTEA Cloud Platform high level architecture

QuantumTEA service

- validate the the user request (multiple circuits support)
- estimate the resources (RAM) needed by the docker container
- instantiate containers from our QuantumTEA docker image on K8s nodes (i.e. CloudVeneto Vms)
- monitor the execution and collect the results

Hypervisor host 1:

- 2 x Intel(R) Xeon(R) Gold 5320 CPU @ 2.20GHz (26 cores/52 threads each)
- 512 GB RAM
- 1 GPU NVIDIA A30

Hypervisor host 2:

- 2 x Intel(R) Xeon(R) Gold 6230R (26 cores/52 threads each)
- 512 GB RAM
- 1 GPU NVIDIA RTX A4000

QuantumTEA Cloud Platform architecture

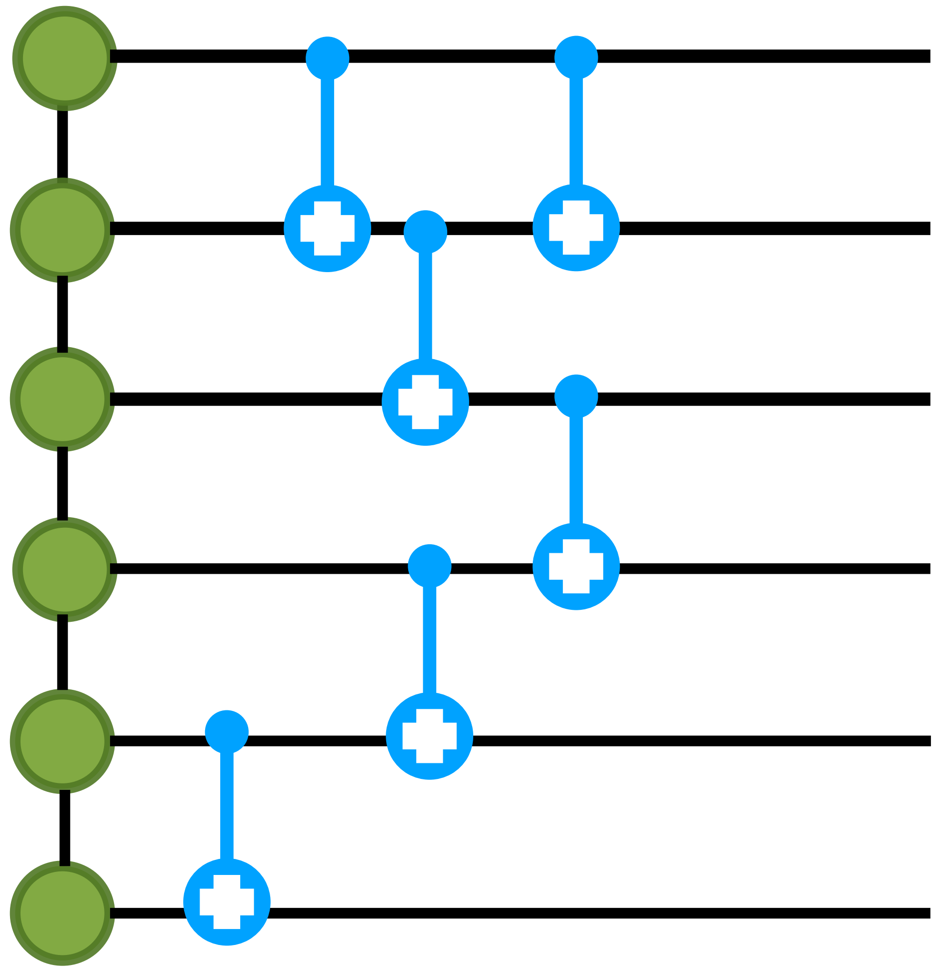
QuantumTEA operator

- Implemented in GO
- QuantumTEAJob Custom Resource Definition (CRD)
- AuthN webhook: added support to IAM

Redis datastore

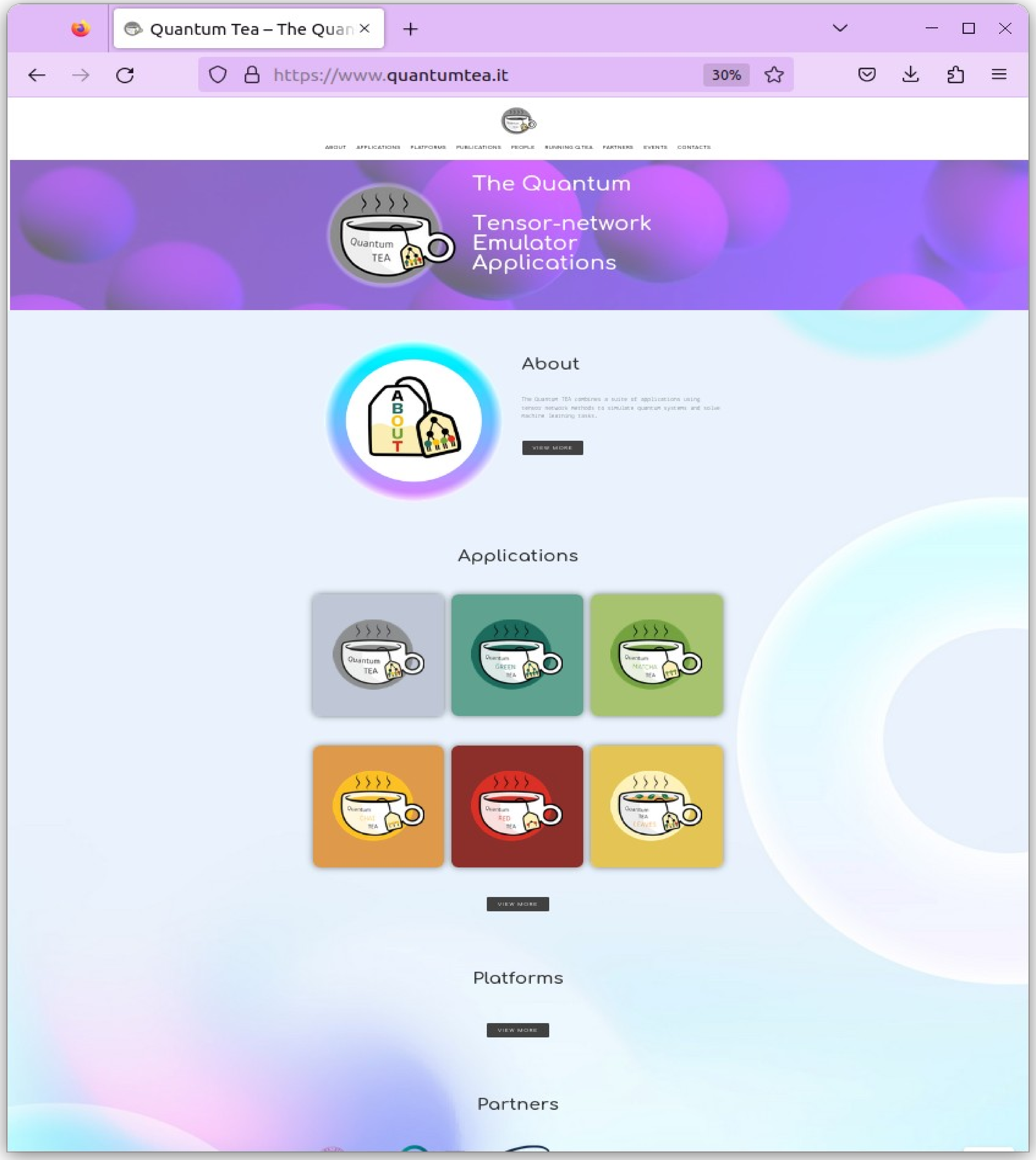
- QuantumCircuit queue (one per job)
- Allow the execution of M circuits by N containers in parallel (M>=N)
- Collect the results

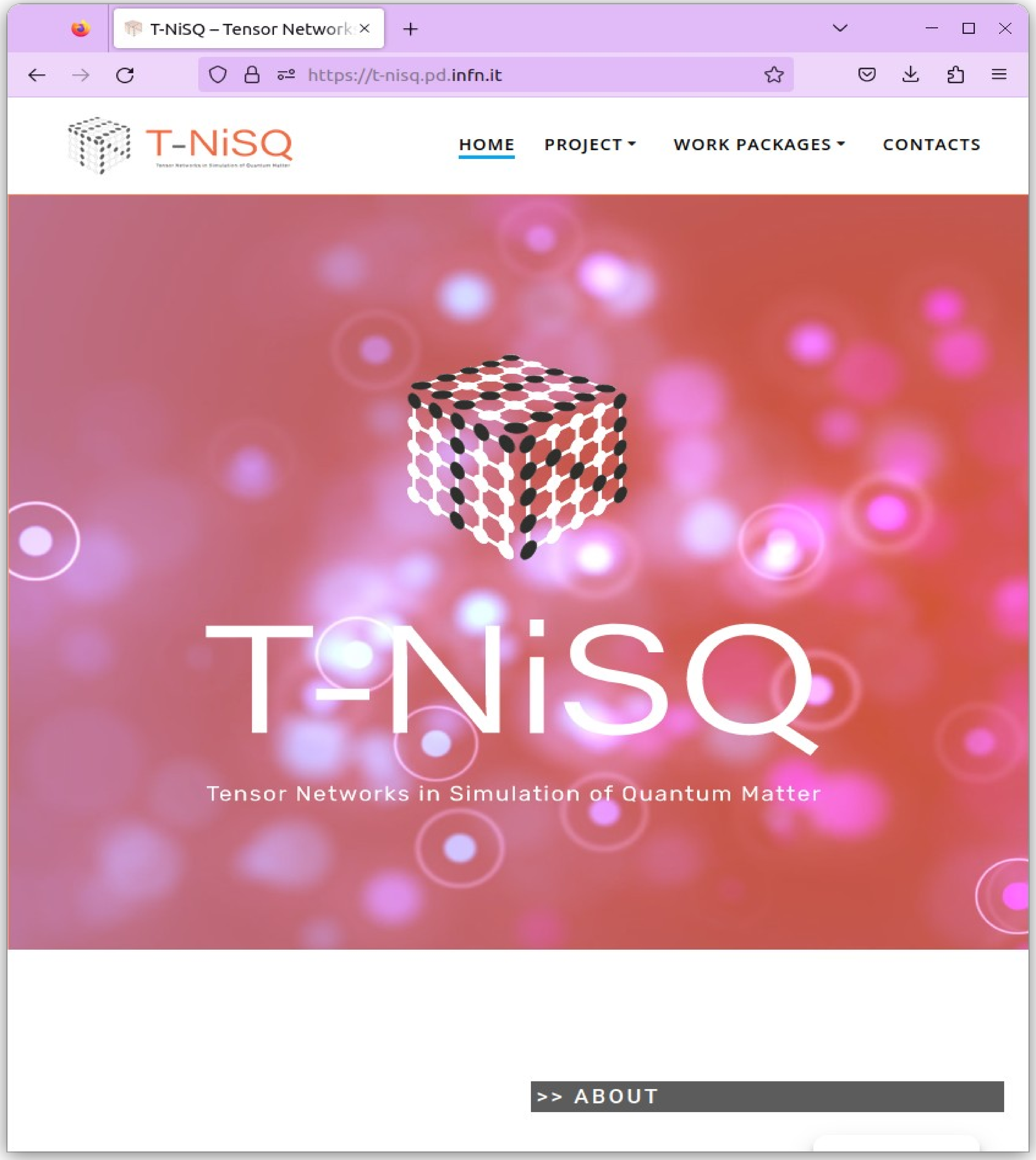
Outlook and next steps



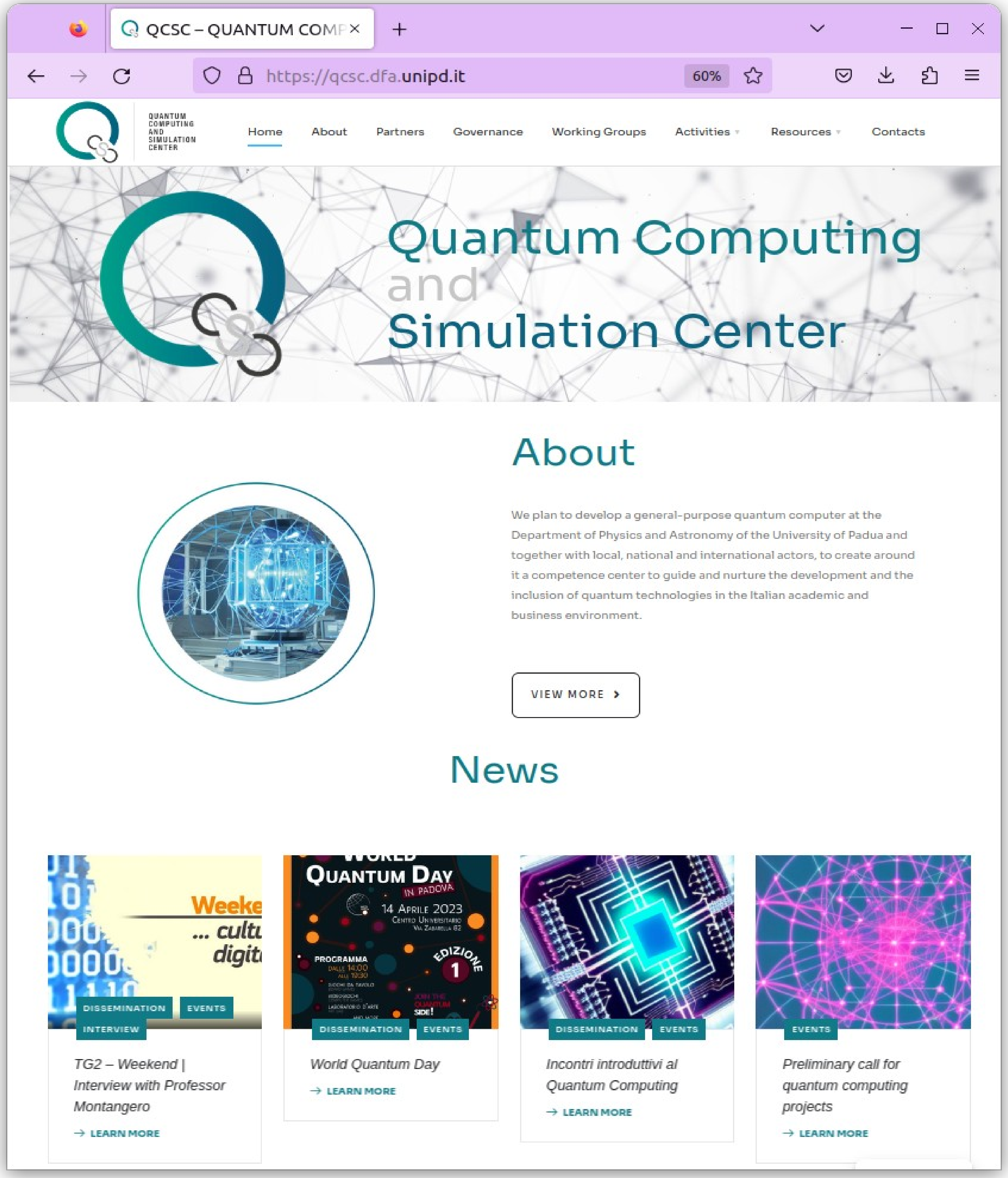
- **QuantumTEA emulator:**
  - New tensor network ansatzes for the simulation;
  - Introduce and emulate noisy processes.
- **QuantumTEA Cloud Platform:**
  - Service not yet available to be tested by external users;
  - To move from prototype to production ready:
    - Improvement the user request validation;
    - simplify getting the token from IAM and test the integration;
    - user guide and units test;
    - stress tests with real use cases (complex circuits).

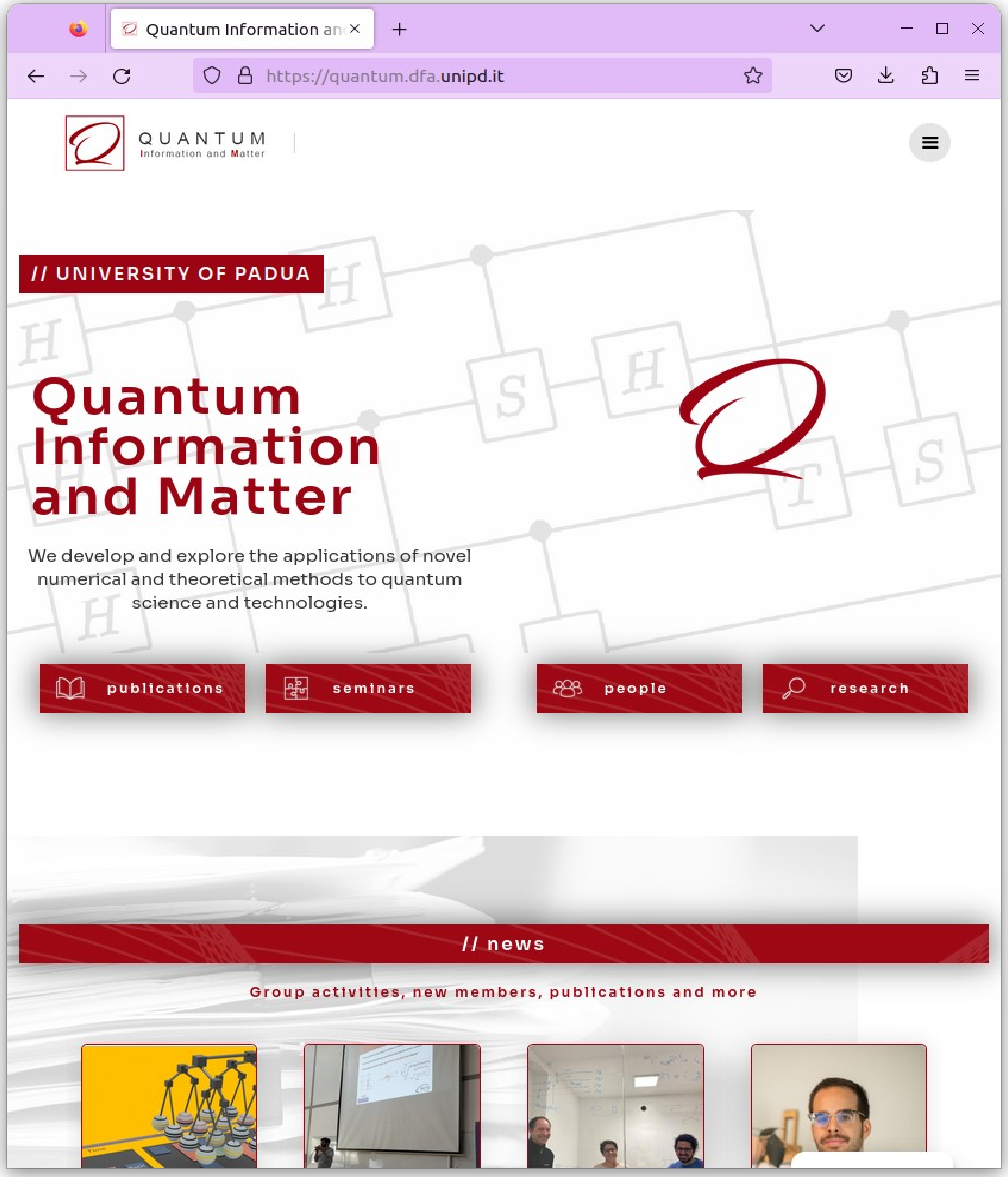
Our websites 1/2



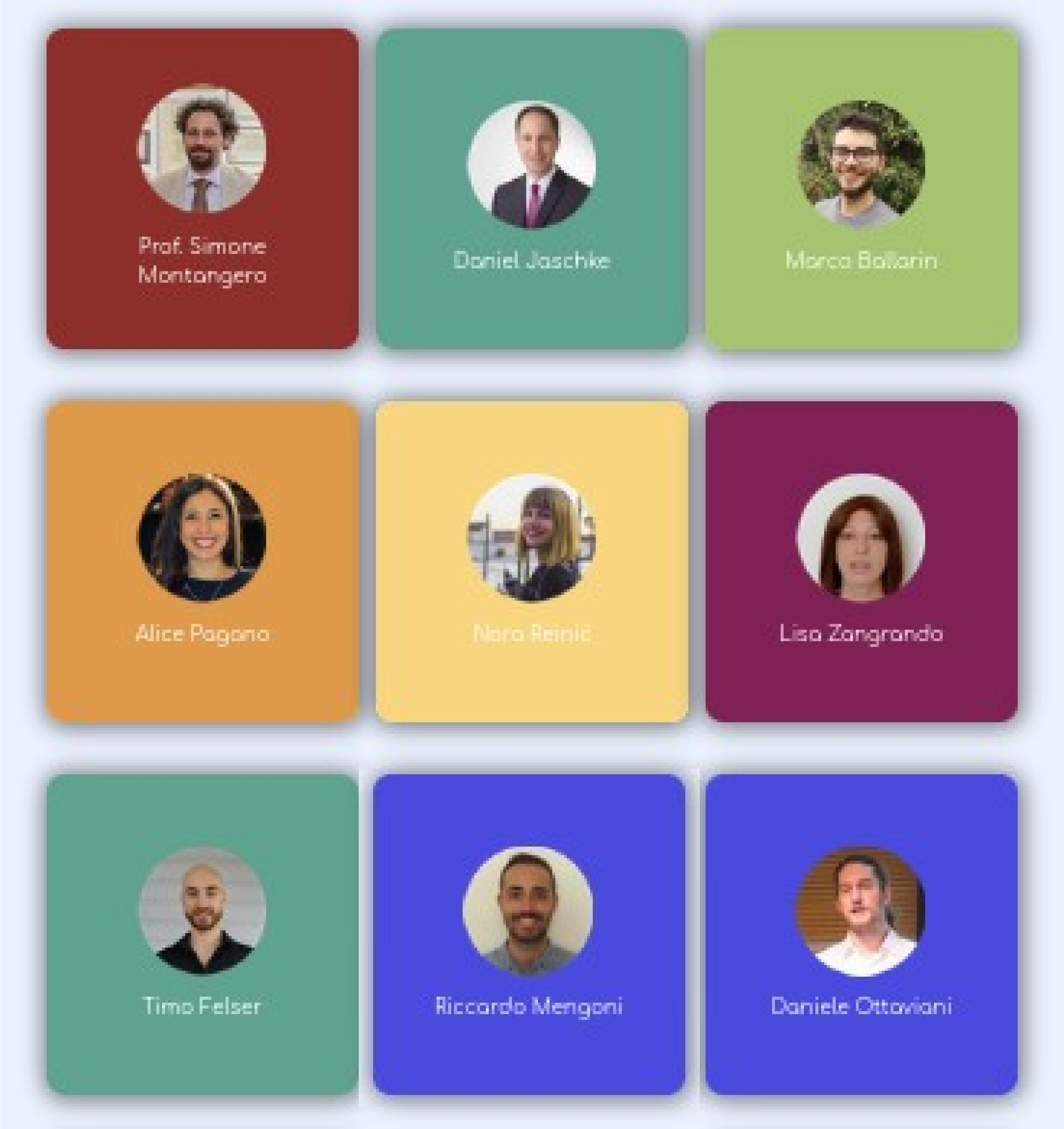


Our websites 2/2





Conclusions



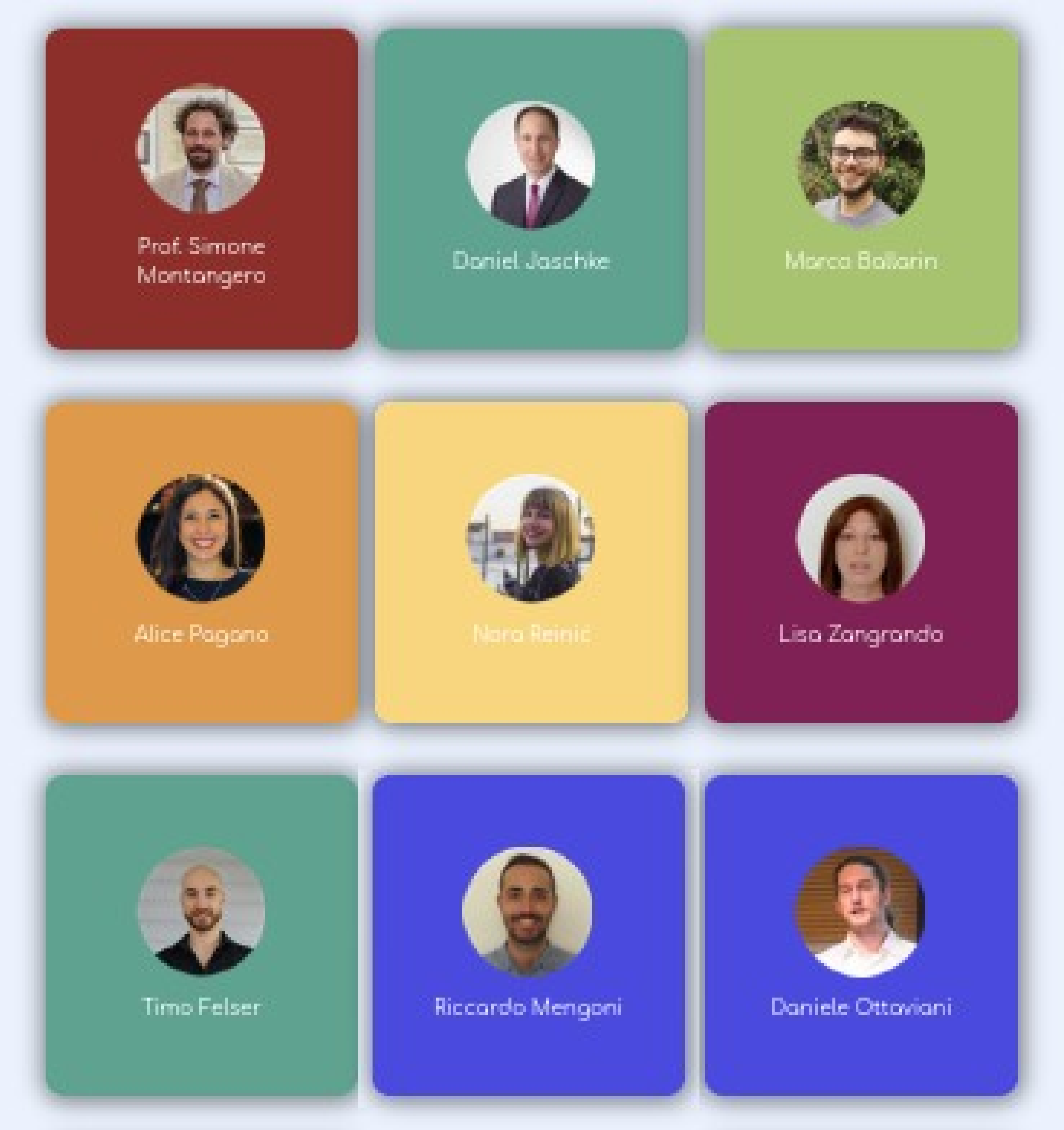
QuantumTEA developers

Documentation and source code on baltig.infn.it

Contact us:

- quantumtea@lists.infn.it
- quantumcomputer@dfa.unipd.it

Thank you!



QuantumTEA developers

Documentation and source code on baltig.infn.it

Thank you !

Contact us:

- quantumtea@lists.infn.it
- quantumcomputer@dfa.unipd.it

Questions

QuantumTEA developers

Documentation and source code on baltig.infn.it

Contact us:

- quantumtea@lists.infn.it
- quantumcomputer@dfa.unipd.it