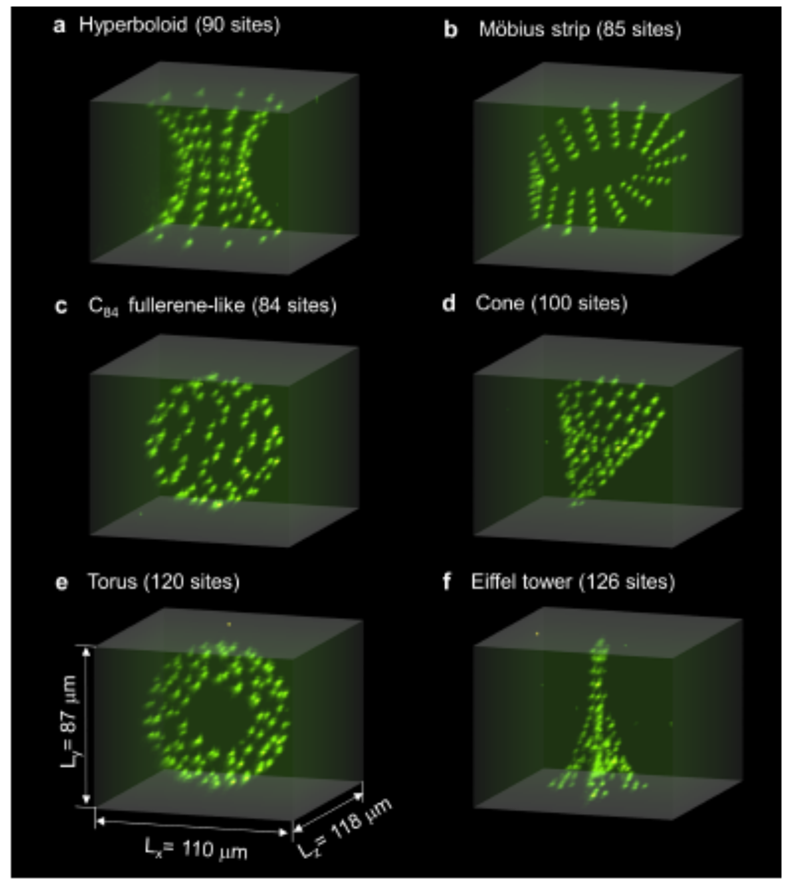
QTCM computational lab on quantum simulation

Lecture 1

- Introduction to Quantum Computing
- Heisenberg XXZ model

Lecture 2

- Quantum Simulation with ultracold atoms
- Bose/Fermi Hubbard model

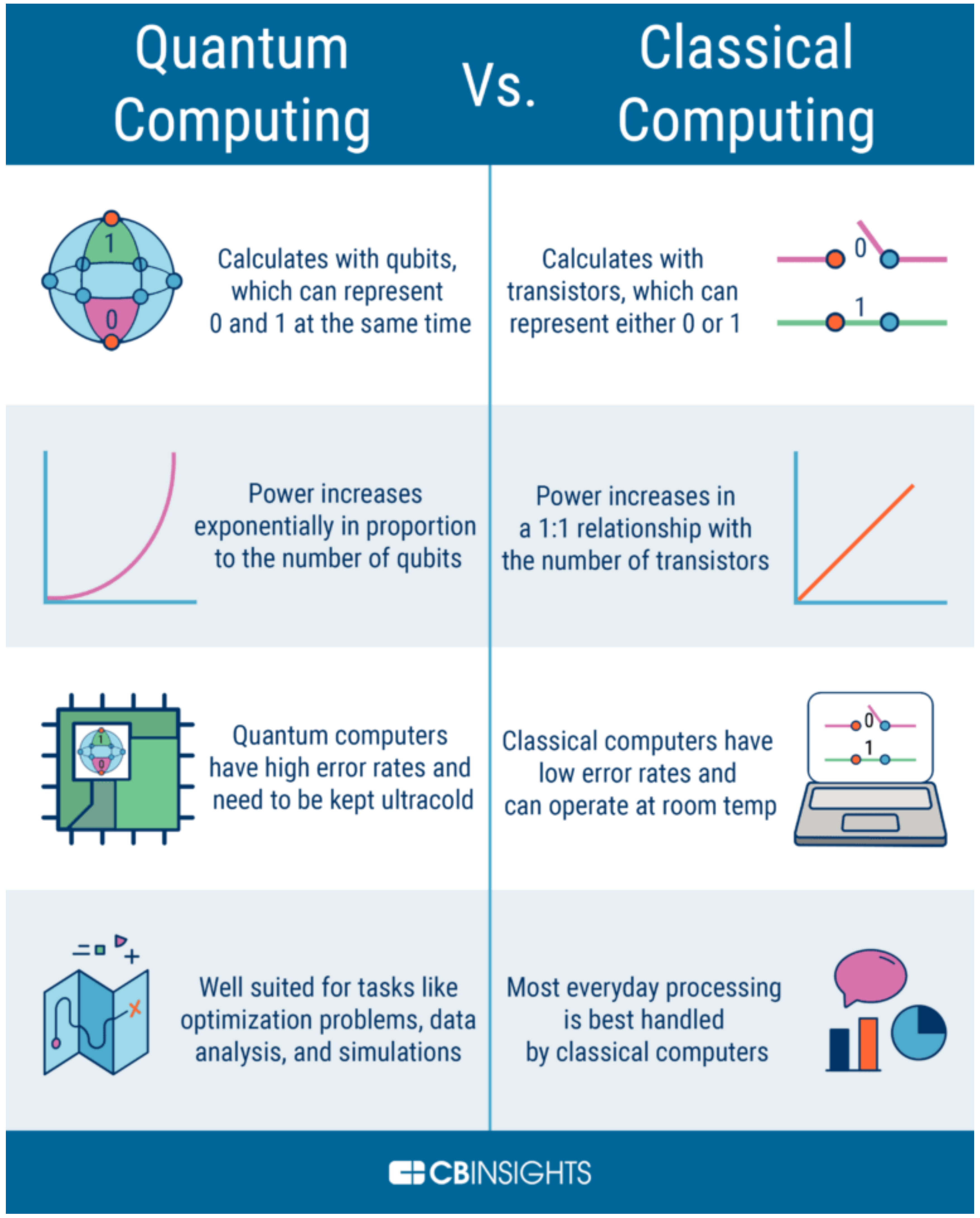


Lecture 3

- Introduction to Topological matter
- SSH model and Haldane phase

<u>Final examination: QuSpin https://quspin.github.io/QuSpin homework project (up to 4 points) to be delivered (at least) 1 week before your oral exam to luca.barbiero@polito.it</u>

QTCM computational lab on quantum simulation



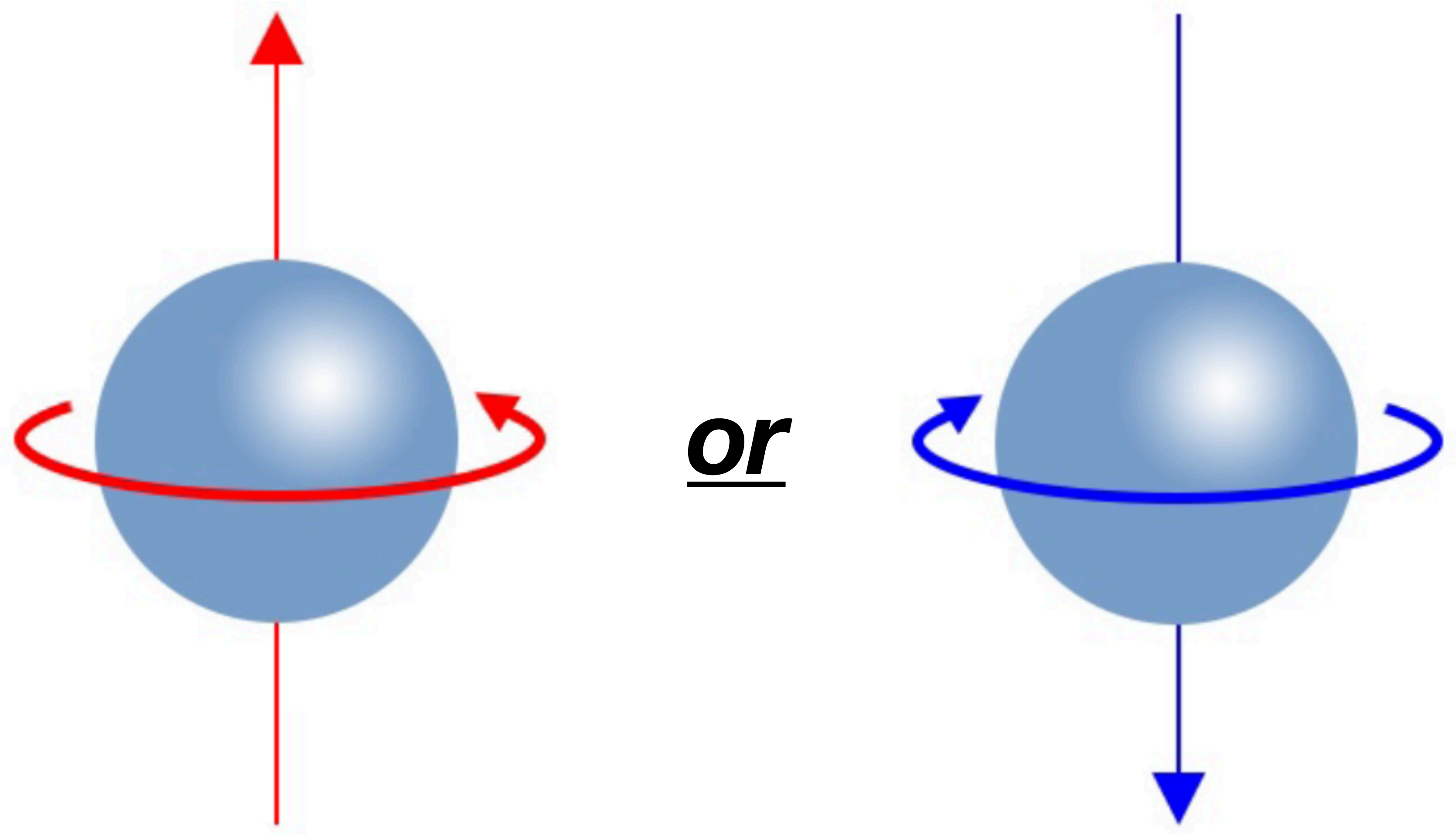
Quantum Computing	Vs.	Classical Computing
Calculates with qubits, which can represent 0 and 1 at the same time		Calculates with transistors, which can represent either 0 or 1
Power increases exponentially in proportion to the number of qubits		Power increases in a 1:1 relationship with the number of transistors
Quantum computers have high error rates and need to be kept ultracold		Classical computers have low error rates and can operate at room temp
Well suited for tasks like optimization problems, data analysis, and simulations		Most everyday processing is best handled by classical computers

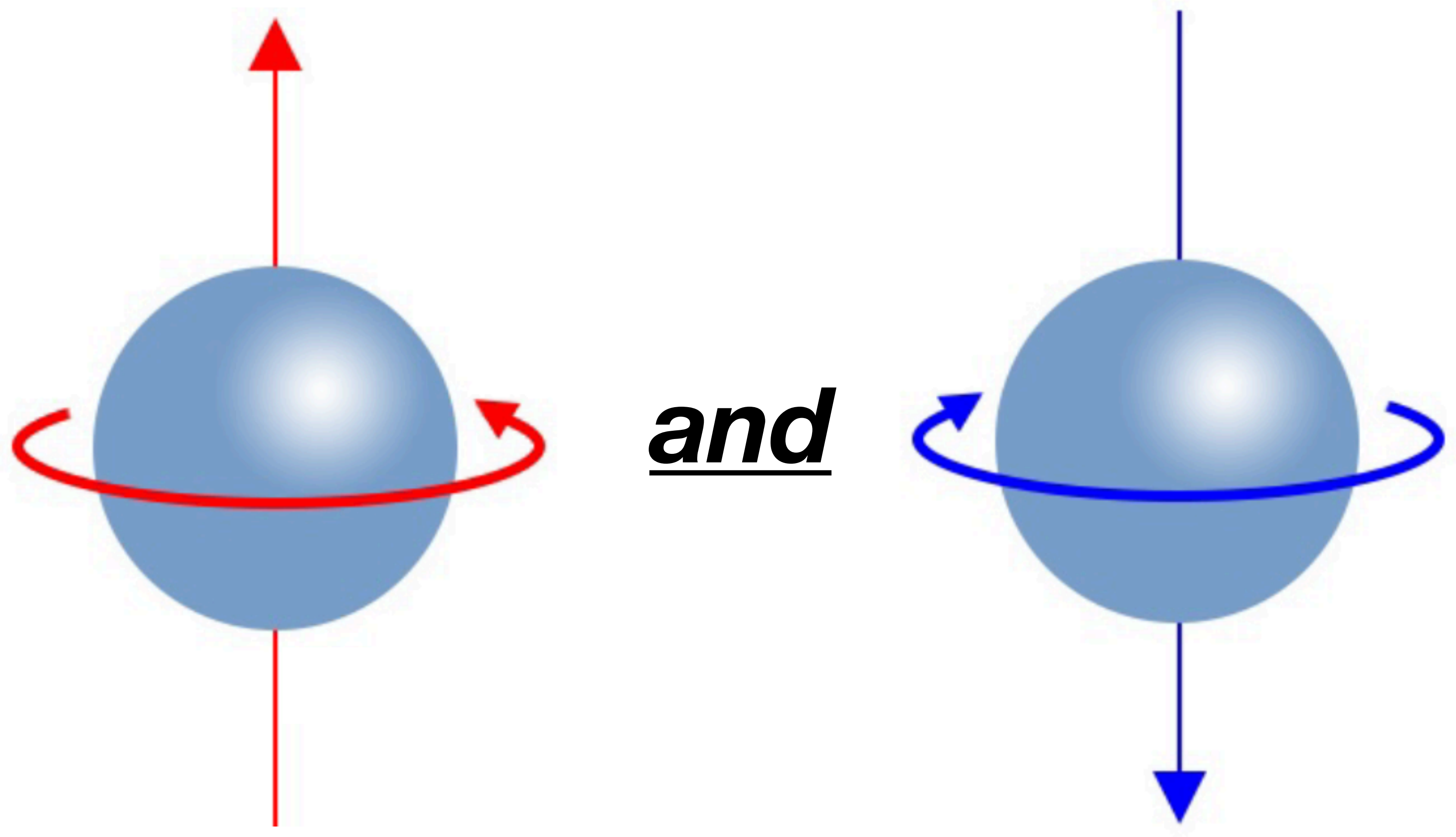
CBINSIGHTS

Politecnico di Torino

QTCM computational lab on quantum simulation

classical ***quantum***





Contrary to classical systems, Quantum "objects" attain for superposition states

$$|\psi\rangle = c_1 |\uparrow\rangle + c_2 |\downarrow\rangle$$

General Framework

Qubits are the elementary carriers of information of a quantum computer. They play the same role as the **bits** stored on the hard drive of your notebook which are processed by its central processing unit (CPU) to make stuff happen (like printing these letters). Similarly, a quantum computer must be able to **store** qubits on a “quantum hard drive” (this is called a **quantum memory**) and to **manipulate** qubits on some sort of “quantum CPU” (a **quantum processor**).

When you think about the bits whizzing around in your notebook, you typically do this in an **abstract way**: they are just “things” with two possible states (1 or 0); you completely ignore what *exactly* these two states are (like: high/low *voltage* in wires, or up/down *magnetization* on the disc of a hard drive). Why? Because it is a useful abstraction for writing programs!

What is a Qubits?

One often hears that “a qubit is like a classical bit that can be 0 and 1 at the same time” – but what does that actually mean?

Let us start with a fancy “***classical***” **bit**. Instead of “0” and “1”, we use the two orthogonal **vectors**

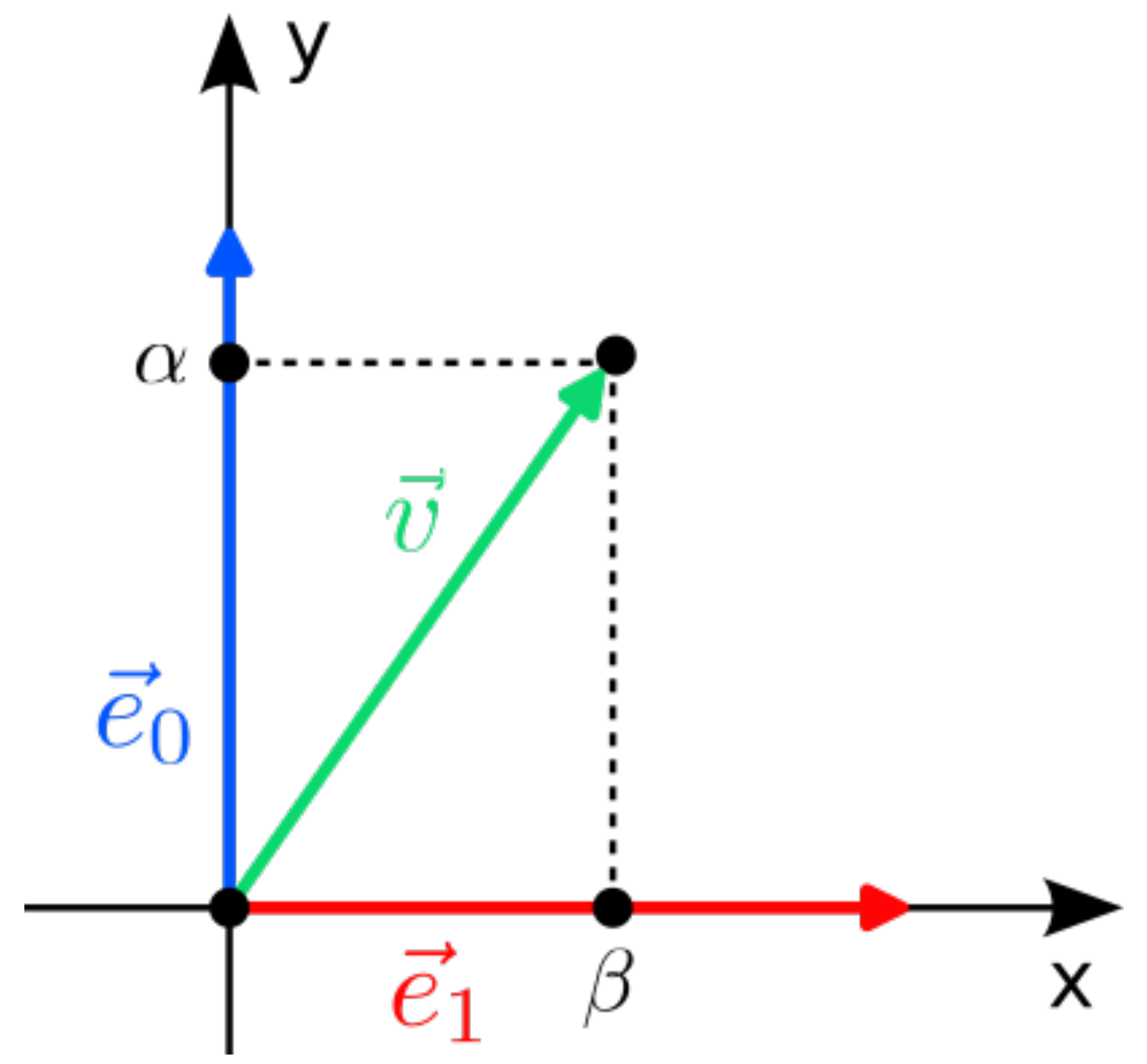
$$0 \leftrightarrow \vec{e}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad 1 \leftrightarrow \vec{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

we would like to come up with a thing that can be *in two states at once*. There is no natural way to combine the discrete states “0” and “1” to something “in between”; but there *is* for vectors: We can scale and add vectors to form new vectors, a procedure called **linear combination**!

What is a Qubits?

The most general linear combination of our two “bit vectors” is the vector

$$\vec{v} = \alpha \cdot \vec{e}_0 + \beta \cdot \vec{e}_1 = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

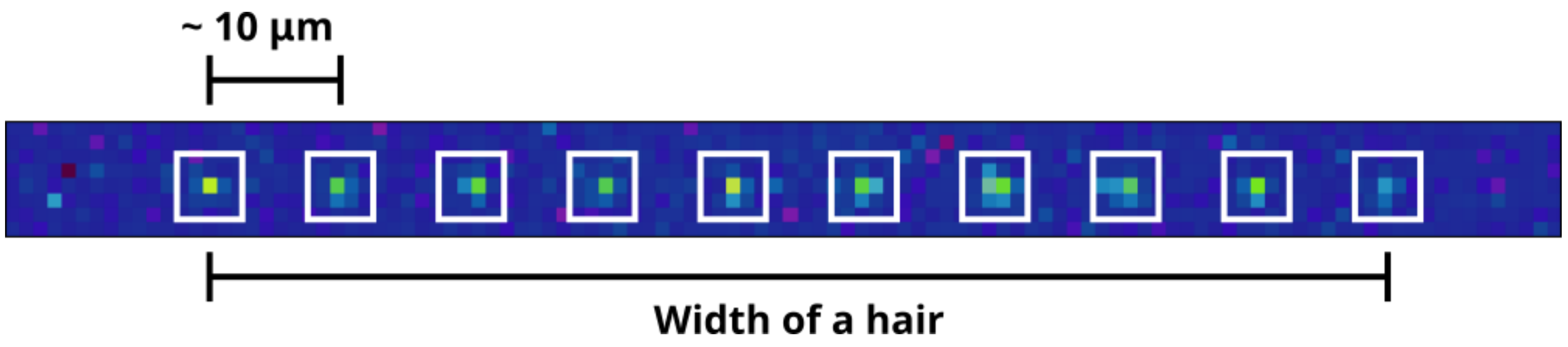


with arbitrary real coefficients $\alpha, \beta \in \mathbb{R}$ that are called **amplitudes** in quantum mechanics. For $\alpha = 1$ and $\beta = 0$ it is $\vec{v} = \vec{e}_0$ (the “0”), while for $\alpha = 0$ and $\beta = 1$ it is (the “1”). But for generic β and α, $\vec{v}$ is neither $\vec{e}_0$ nor $\vec{e}_1$. We can therefore put forward the following hypothesis:

The state of a single qubit is described by a two-dimensional vector $\vec{v}$.

Example of a “physical” Qubit

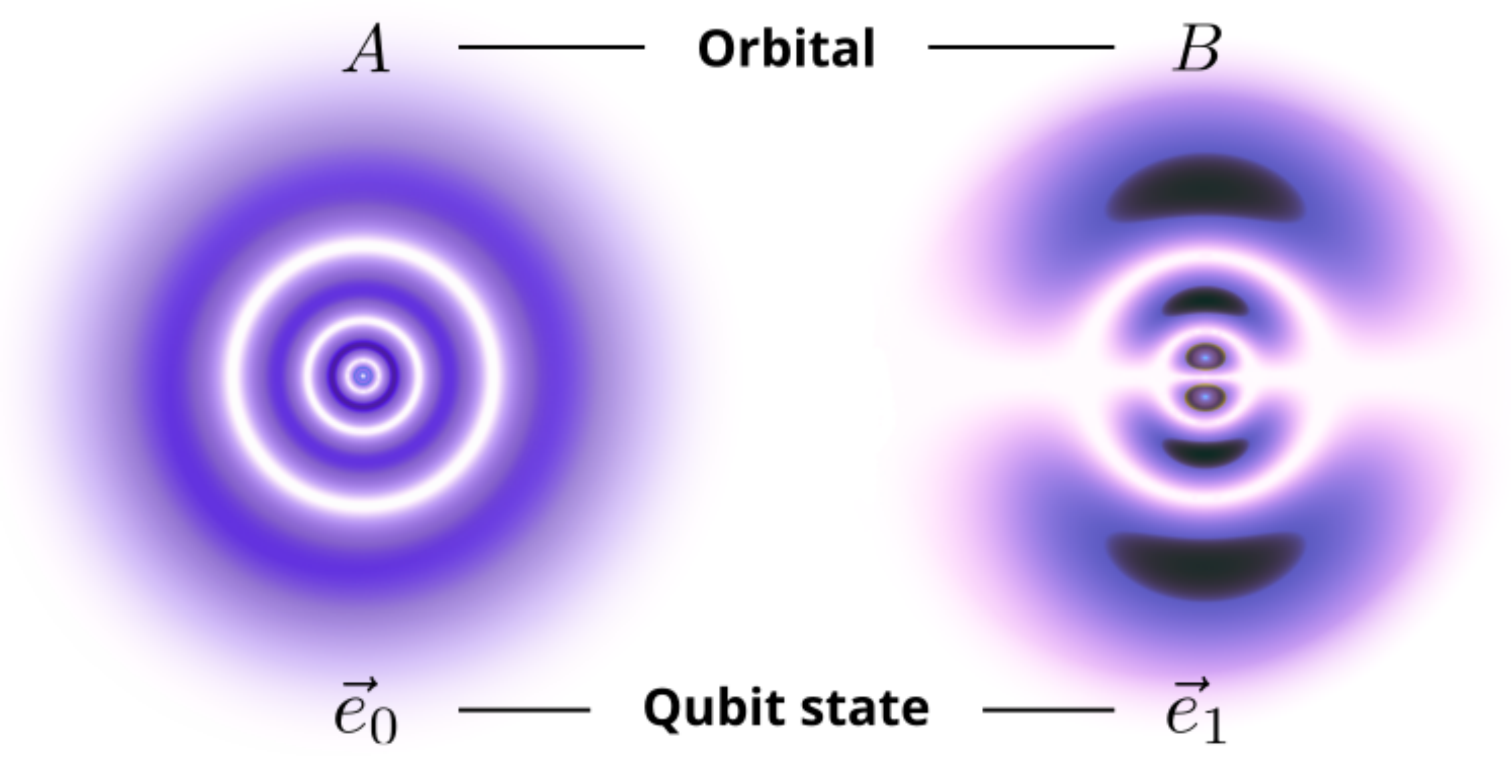
ATOMS!!!! cooled down and then trapped by lasers. If we illuminate these atoms with another laser of a specific wavelength (= color), the atoms start to glow (they *fluoresce*). This glowing can be detected by a very sensitive digital camera that observes the trapped atoms through a microscope. A photograph of 10 atoms arranged in a chain looks like this:



But what are the two states? many possibilities….

Example of a "physical" Qubit: orbitals

From chemistry you know that the electrons of an atom all live in a discrete set of *orbitals*. They are the quantum mechanically correct description of the energy levels in the simpler Bohr model. Their existence and shape are a direct consequence of quantum mechanics, but here we simply want to use them as our two states! Thus we just pick two A and B



An atom with electrons in the orbital A then corresponds to qubit in the state $\vec{e}_0$ whereas atom with electrons in the orbital B corresponds to qubit in the state $\vec{e}_1$. Because the electrons obey the laws of quantum mechanics, the state of an atom can be a linear combination of $\vec{e}_0$ and $\vec{e}_1$, i.e. a superposition state!

Why single atoms?

Now you know how we realize qubits in our laboratory. But why single atoms, you might ask? Isn't it very complicated to work with single atoms? (It is!) Why not larger objects that are easier to control? This is a good question and it has to do with our next topic: *measurements* as in quantum mechanics obtaining information about the state of a qubit *changes the state itself*. But this means that if something (say, an air molecule or a single photon) bumps into your qubit and carries away information about its state, this interaction perturbs the qubit. For a qubit to be useful, one must therefore be able to shield it almost perfectly from all possible environmental influences. This shielding is much easier for atoms in vacuum than for larger ("macroscopic") objects that very easily spread information about their state into the environment. So yes, controlling single atoms is hard, but we are rewarded with qubits that preserve their quantum states long enough so that one can hope to do useful computations with them.

Measurements

The quantumness enters the stage when we **measure** the qubit. In classical physics, there is a one-to-one correspondence between *states* and *observations*: systems in different states look differently when we measure them. At first, this statement seems tautological: isn't it the sole purpose of different states to describe the different observations we can make? In classical physics, yes; in quantum mechanics, no! When you measure a qubit, it does not look like a compass needle pointing in the direction $\vec{v}$ it looks like a classical bit pointing in either $\vec{e}_0$ or $\vec{e}_1$ and nothing in-between. The **probabilities** of either result depend on the state $\vec{v}$ and, as it turns out, are given by the **squares of the amplitudes** of the vector (**BORN RULE**):

$$\vec{v} = \alpha\,\vec{e}_0 + \beta\,\vec{e}_1 \xrightarrow{\text{Measurement}} \vec{v}_{\text{new}} = \begin{cases} \vec{e}_0 & \text{with probability } p_0 = \alpha^2 \\ \vec{e}_1 & \text{with probability } p_1 = \beta^2 \end{cases}$$

Here, $\vec{v}_{new}$ denotes the new state of the qubit *after* the measurement. Notice that as α and β are interpret as probabilities, the relation $\alpha^2 + \beta^2 = 1 = |\vec{v}|^2$ holds

Measurement rules:

Measurements in quantum mechanics are described by the **Born rule** according to which a qubit is measured in one of two **discrete states** with probabilities that equal the **squares of the amplitudes** of its state vector.

State vectors must be **normalized** because we interpret the squares of their amplitudes as probabilities.

Measurements in quantum mechanics modify the state of the object measured. This is called **wave function collapse**.

The measurement process makes quantum mechanic a **probabilistic theory.** For quantum computing this implies that algorithms must be run several times to compute **averages**.

The randomness of measurement outcomes is *not* due to a lack of knowledge of "hidden parameters".

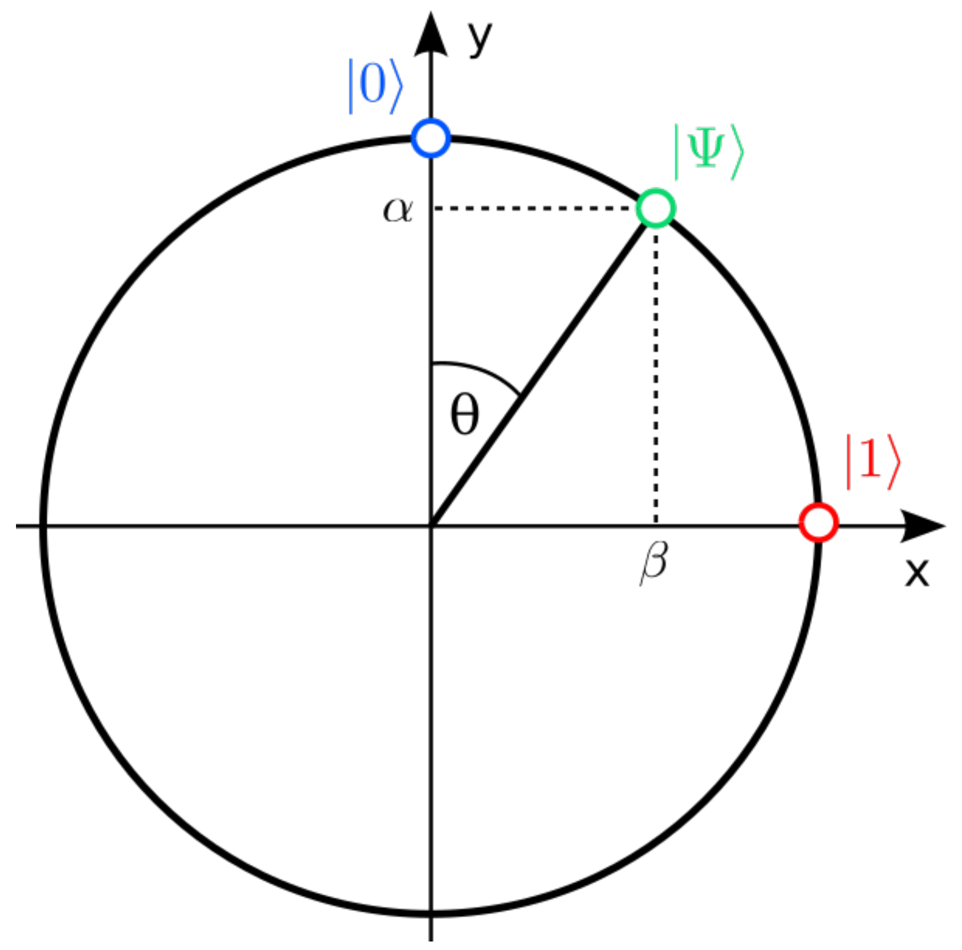
Quantum Physics Notation

If you previously encountered a text on quantum mechanics, you may wonder why none of the previous formula *look* familiar…This is the usual notation

$$\vec{v} \rightarrow |\Psi\rangle \qquad \vec{e}_0 \rightarrow |0\rangle \qquad \vec{e}_1 \rightarrow |1\rangle$$

The linear combination then looks like $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha^2 + \beta^2 = 1$

This is the representation for a **quantum state** (or **wave function**) of a single **qubit**. The numbers α and β describe the state completely and are called **(probability) amplitudes**

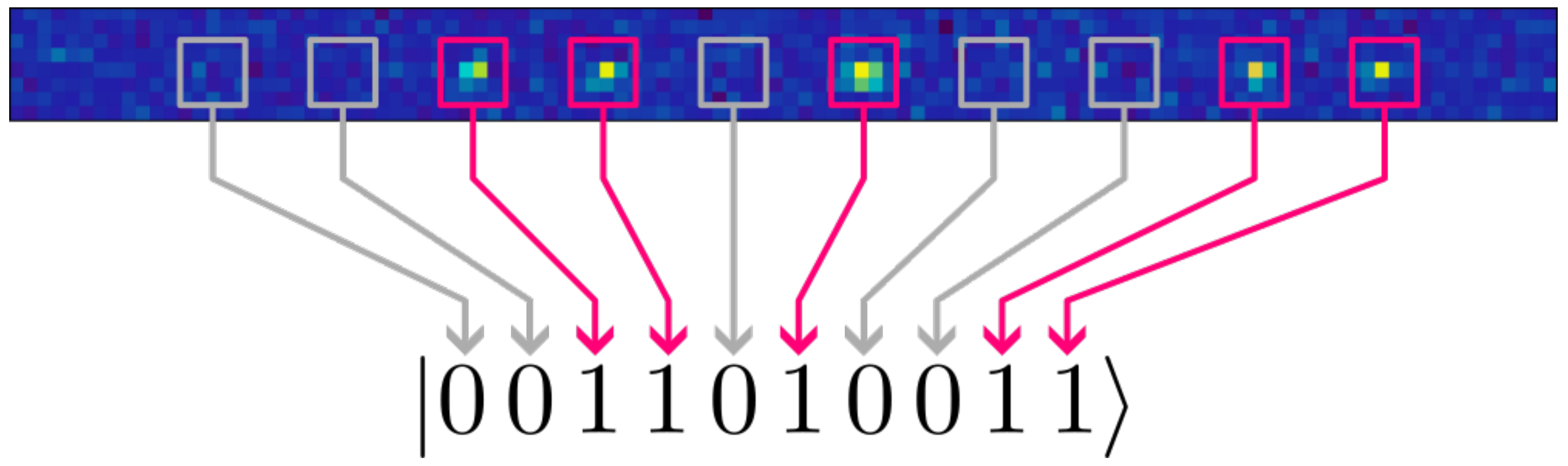


Many Qubits

We should expect that if we measure N qubits, we will observe a result that looks like N *bits*, where each bit can be either instate "0" or in state "1". We write such a quantum state (vector!) where N qubits are in the configurations $x_i \in \{0,1\}$ (i labels the qubits) as $|x_1 x_2 \dots x_N\rangle$

Let us consider an example: If we have only N=2 qubits and measure them, we will find one of the $2^N = 4$ states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$

$$|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \qquad |\Psi\rangle \xrightarrow{\text{Measurement}} |\Psi_{\text{new}}\rangle = \begin{cases} |00\rangle & \text{with probability } p_0 = \alpha^2 \\ |01\rangle & \text{with probability } p_1 = \beta^2 \\ |10\rangle & \text{with probability } p_2 = \gamma^2 \\ |11\rangle & \text{with probability } p_3 = \delta^2 \end{cases}$$



Concrete Numbers

Let us assume we have a *classical* computer (right now that's all we have anyway) and would like to *simulate* how the quantum state of N qubits evolves over time. To do this, we have to store the quantum state somehow i.e. α and β. Each amplitude is a real number that we can store in our classical computer approximately. In a programming language we would use a variable for this purpose for which your computer typically reserves 4 bytes of your random access memory (RAM).

How much memory M do we need to store all amplitudes of the quantum state of N qubits? Easy

$$M = 2^N \times 4byte$$

So for 2 qubits we need as little as M=16 byte of memory. Not so impressive….

Concrete Numbers

How much qubits do you think you can simulate on your computer that has, let's say, around M=16 Gigabyte of RAM?

$$N = \log_2(M/4\,\mathrm{byte}) = \log_2(16 \cdot 10^9/4) \approx 32$$

Not so impressive…Suppose we have M=1Terabyte=1000Gigabyte of RAM (and cost up to 10.000€!)

$$N = \log_2(M/4\,\mathrm{byte}) = \log_2(1000 \cdot 10^9/4) \approx 38$$

Not so impressive (again)…The number of amplitudes that define a quantum state, and therefore the amount of memory needed to simulate such a state on a classical computer, grows (better: explodes) exponentially with the number of qubits.

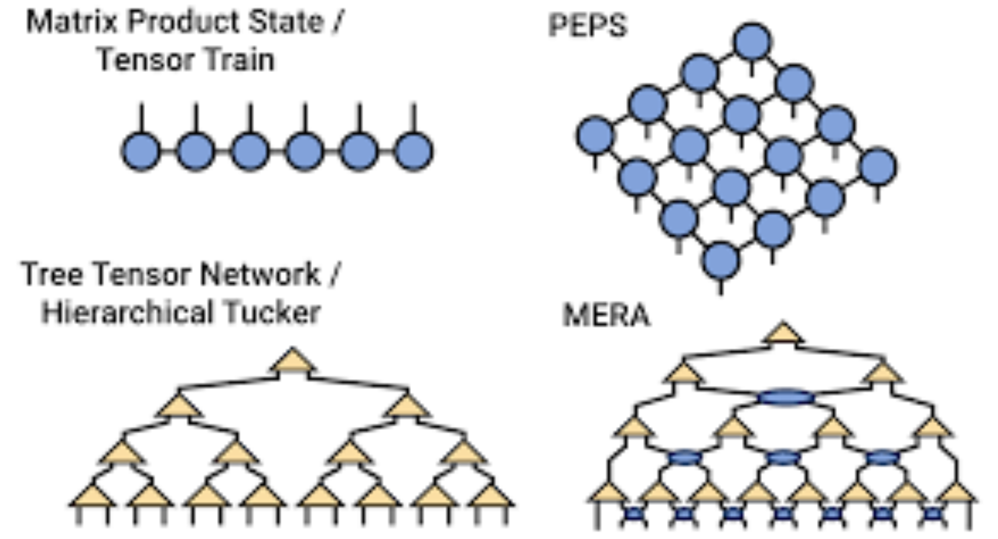
N=100 Qubit…..

$$M = 2^{100} \times 4\,\mathrm{byte} = 5070602400912917605\,\mathrm{Terabyte}$$

in the next decades such RAM will never exist….

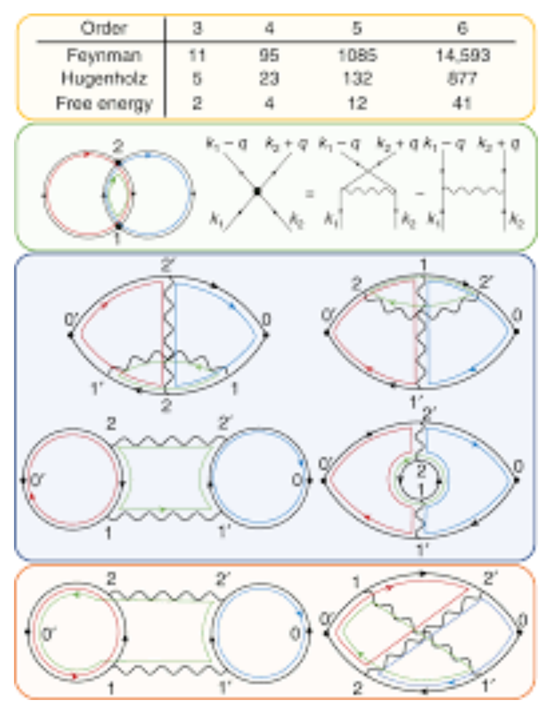
Quantum algorithms working on classical computers

Tensor Networks


Matrix Product State / Tensor Train
PEPS
Tree Tensor Network / Hierarchical Tucker
MERA

GS physics, 1D and 2D systems, Fermi/Bose, T=0 and T≠0, equilibrium and out-of-equilibrium, large systems

Quantum Monte Carlo

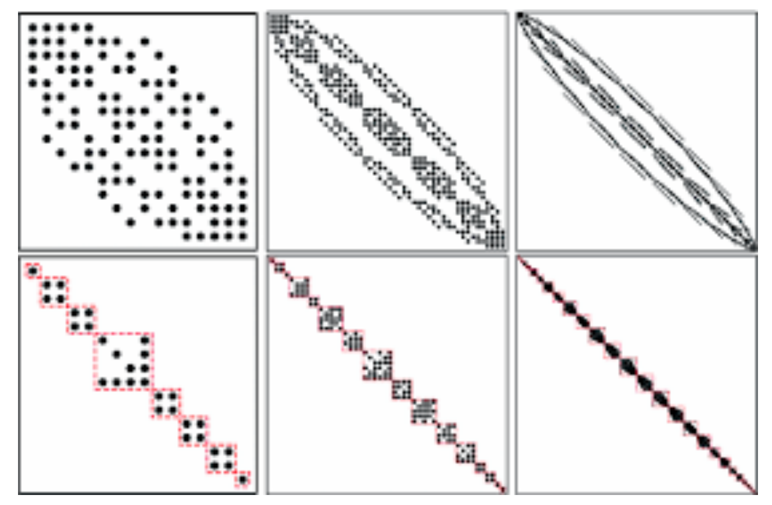


Order	3	4	5	6
Feynman	11	95	1085	14,593
Hugenholz	5	23	132	877
Free energy	2	4	12	41

GS physics, 1D, 2D, 3D systems, Bose (Fermi with problems), T=0 and T≠0, equilibrium and out-of-equilibrium, large systems

Exact Diagonalization

(QuSpin)



Full spectrum, 1D (2D, 3D very limited) systems, Bose/Fermi, T=0 and T≠0, equilibrium and out-of-equilibrium, very small systems

Atomic Quantum Simulators

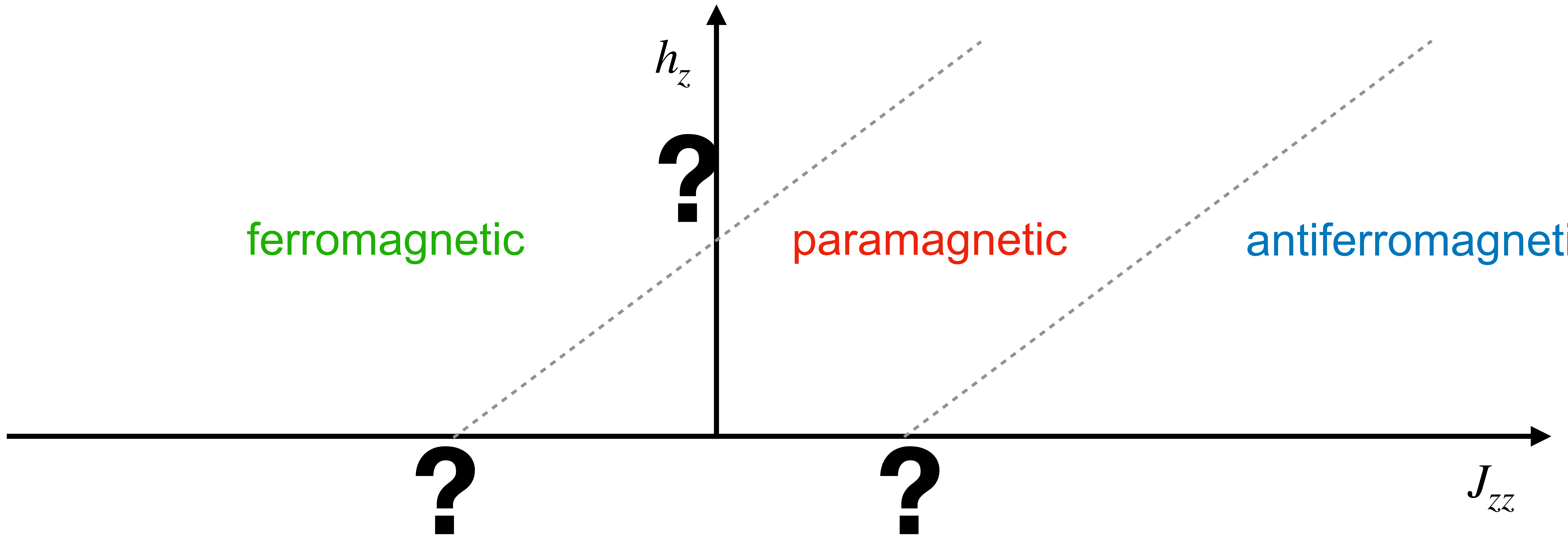
(Lecture 2)

Hamiltonian of N entangled qubit: the XXZ Heisenberg model

$$H = \sum_{j=0}^{L-2} [J_{xy}(S_j^x S_{j+i}^x + S_j^y S_{j+i}^y) + J_{zz} S_j^z S_{j+i}^z] + h_z \sum_j S_j^z = \sum_{j=0}^{L-2} [\frac{J_{xy}}{2}(S_j^+ S_{j+i}^- + S_j^- S_{j+i}^+) + J_{zz} S_j^z S_{j+i}^z] + h_z \sum_j S_j^z$$

$$S_i^z = \frac{\hbar}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad S_i^x = \frac{\hbar}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad S_i^y = \frac{\hbar}{2}\begin{pmatrix} 0 & -\imath \\ \imath & 0 \end{pmatrix} \qquad S_i^\pm = S_i^x \pm \imath S_i^y$$

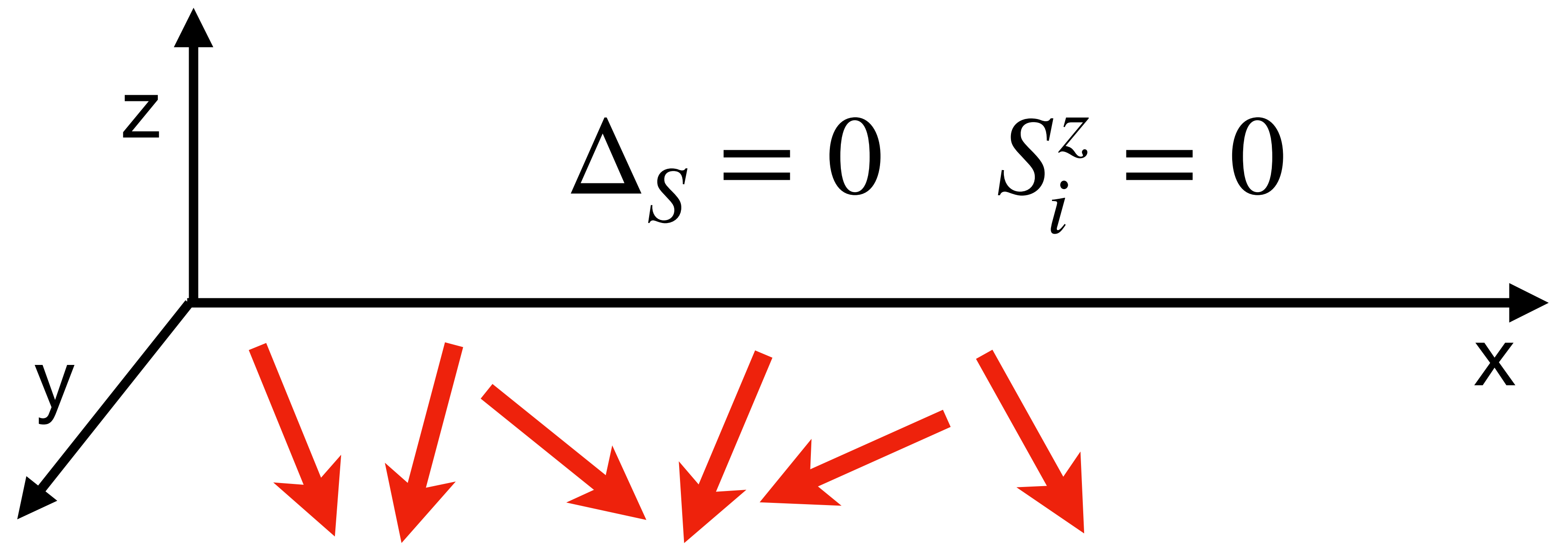
Phase diagram of the XXZ Heisenberg model ($J_{xy} = 1$)



QTCM computational lab on quantum simulation

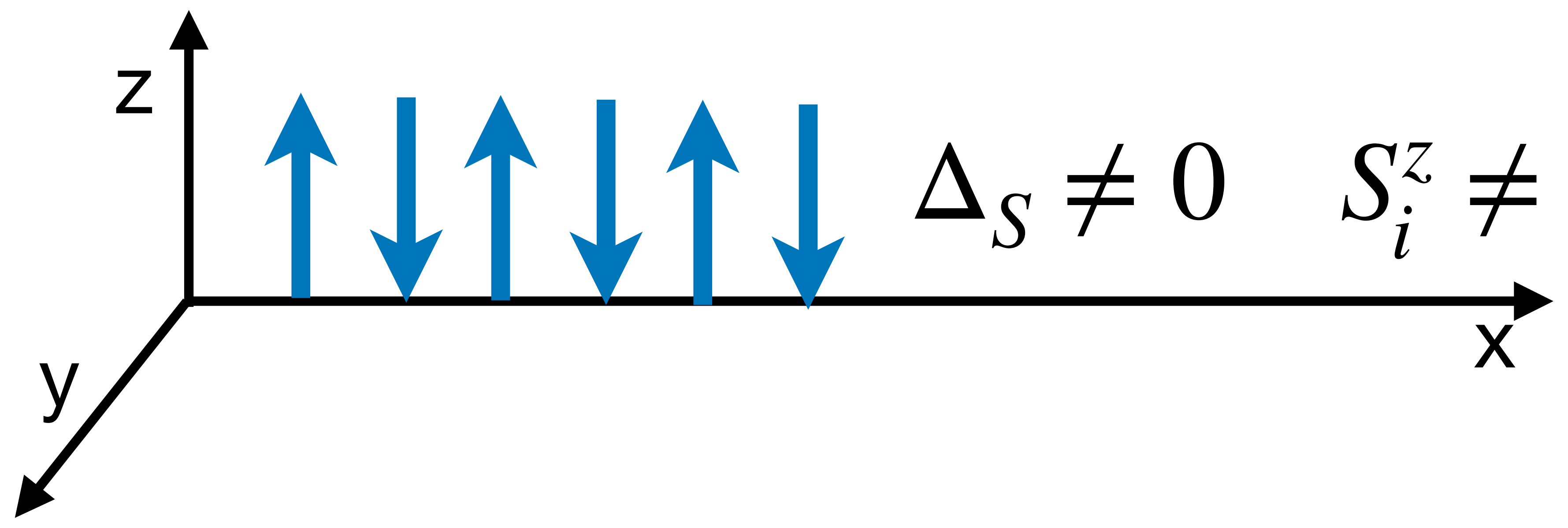
Gap analysis: $\Delta_S(N) = E_{GS}(N, S^z_{tot} = 1) - E_{GS}(N, S^z_{tot} = 0)$ **with** $S^z_{tot} = \sum_{j=1}^{N} S^z_j$

paramagnetic



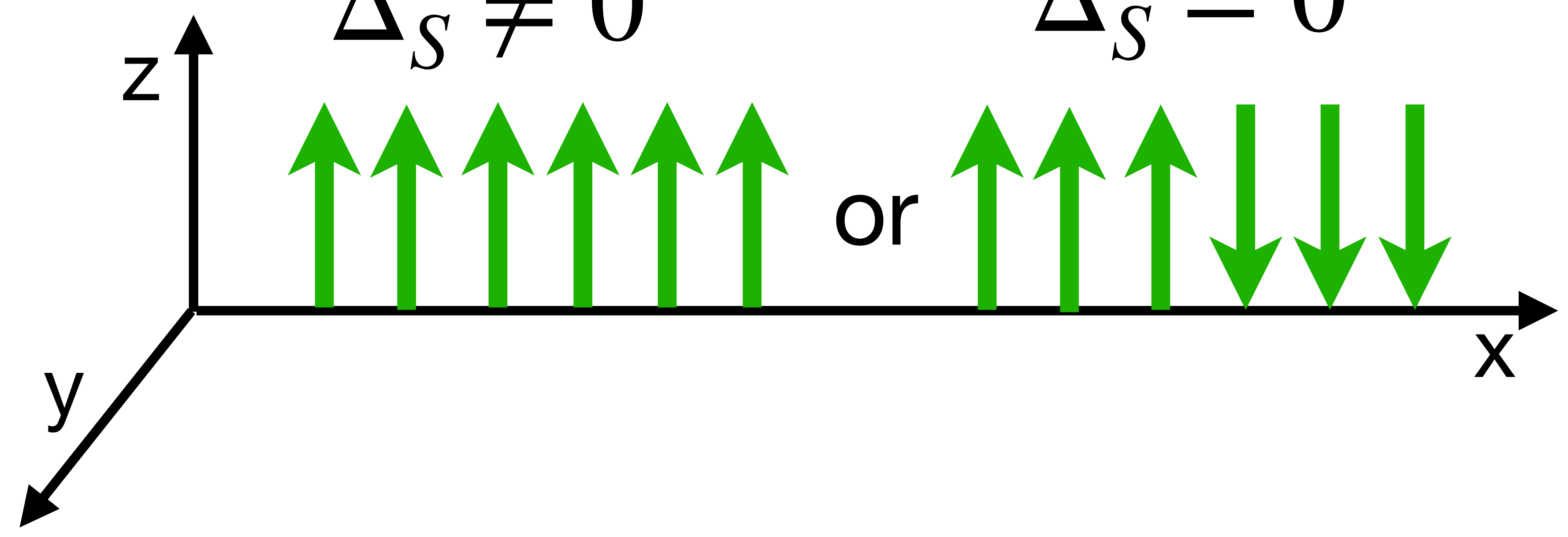
$\Delta_S = 0 \quad S^z_i = 0$

antiferromagnetic



$\Delta_S \neq 0 \quad S^z_i \neq 0$

ferromagnetic

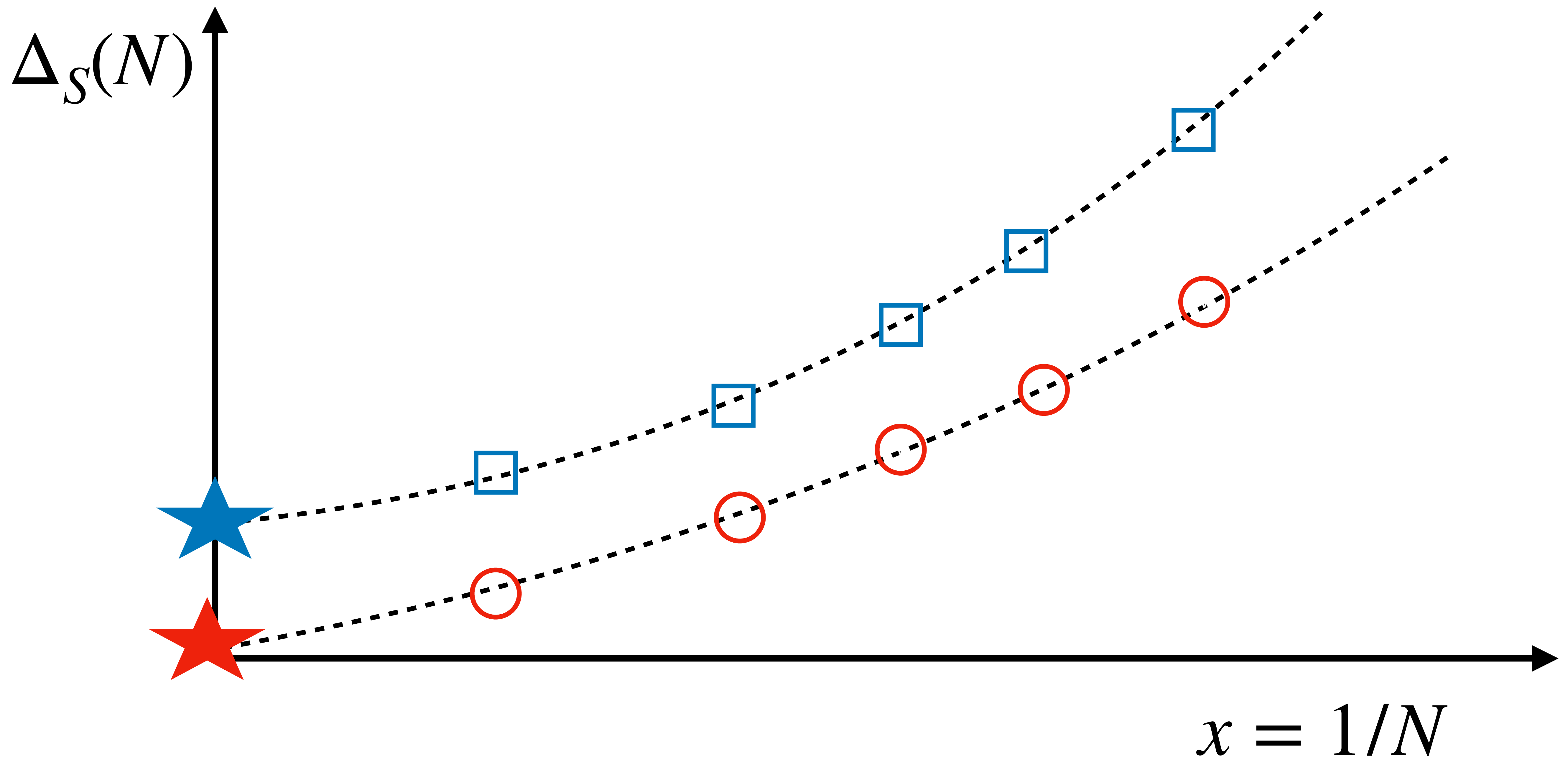


$\Delta_S \neq 0$ or $\Delta_S = 0$

both $S^z_i \neq 0$

Gap extrapolation in the thermodynamic limit

1. calculate $\Delta_S(N)$ for different N
2. report these values on graph with axis $x = 1/N$ and $y = \Delta_S(N)$
3. fit these values using the fitting equation $f(x) = a + b * x + c * x^2$
4. the value $a = \Delta_S(N = \infty)$ corresponds to the gap in the thermodynamic limit


$\Delta_S(N)$
$x = 1/N$