Introduction to quantum computing

Quantum Computing Minicourse ICTP-SAIFR

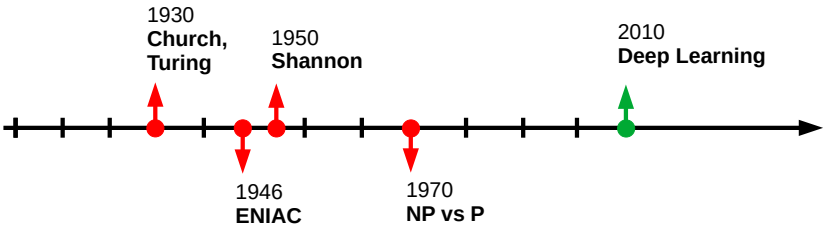
Stefano Carrazza[‡] and Matteo Robbiati[†]

8 April 2024

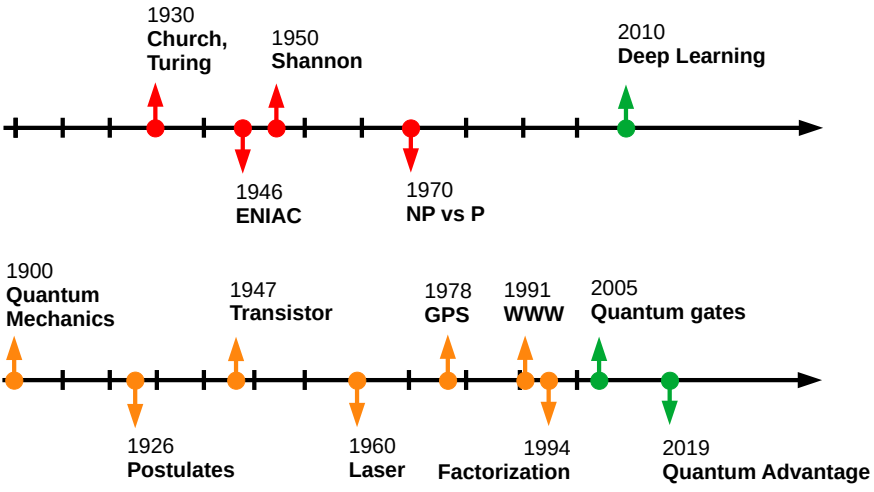
[‡] Associate Professor & Researcher, University of Milan and INFN Milan, Italy.
[†] PhD candidate, University of Milan, Italy and CERN, Switzerland.

Introduction

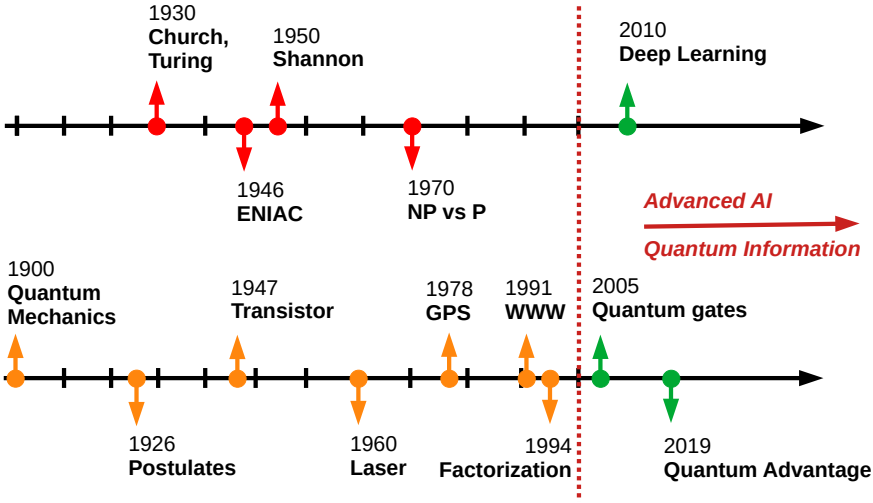
The Quantum Disruption



The Quantum Disruption



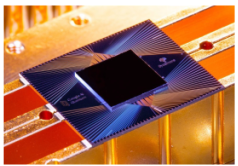
The Quantum Disruption



R&D and adoption of new technologies

Scientific community is moving towards new technologies, in particular **hardware accelerators**:

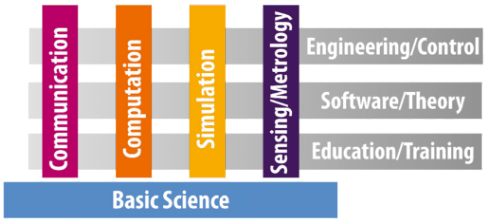
CPU **GPU** **FPGA/ASIC** **Quantum chip**



Moving from **general purpose devices** ⇒ **application specific**

Quantum research

Structure of research field in quantum technologies:

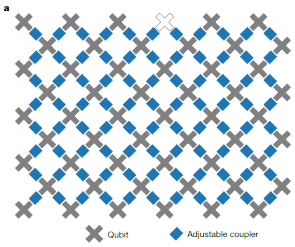


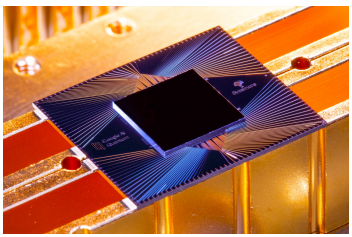
Quantum computing is a paradigm that exploits quantum mechanical properties of matter in order to perform calculations.

$\Rightarrow$ Unitary operators, entanglement, superposition, interference, etc.

Quantum advantage

First quantum computation that can not be reproduced on a classical supercomputer from Google, Nature 574, 505-510(2019):

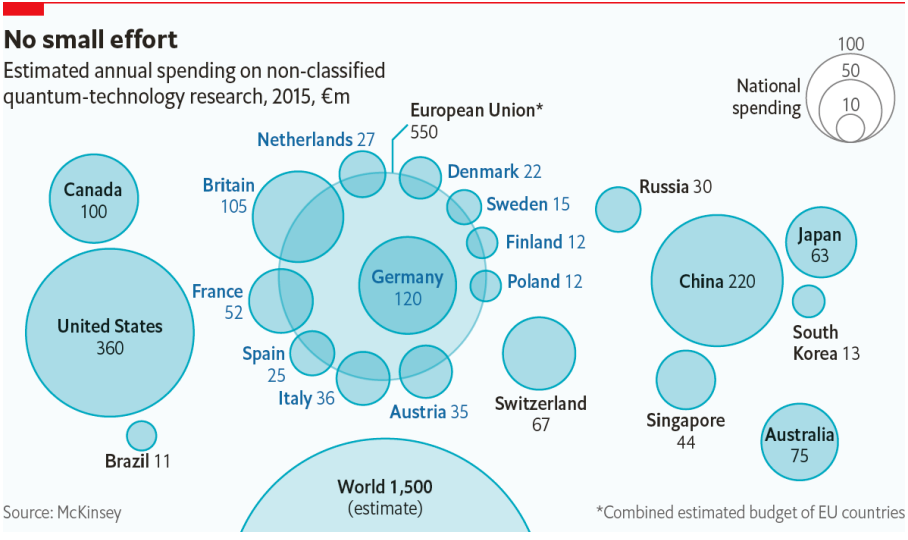




53 qubits (86 qubit-couplers) $\rightarrow$ Task of sampling the output of a pseudo-random quantum circuit (extract probability distribution).

Classically the probability distribution is exponentially more difficult.

Annual spending on quantum technology



Strong synergy between industry and academics worlds

Full-Stack (End-to-End)

Google
IBM
Microsoft
amazon
Honeywell
rigetti
Alibaba Group
D-Wave
XANADU

Software Applications

QCWARE
MULTIVERSE
RIVERLANE
Menten AI
ZAPATA
QUANTUM
QBITLOGIC
QRITHM
QULAB

Cloud Computing

agnostiq
QCWARE
1QBit
BraneCell
Aliro QUANTUM
BLACK HAWK

Quantum Encryption and AI

ISARA
agnostiq
Post-Quantum
SPEQTRAL
INFINITE SHIELD
IDQ
ZY4

Systems & Firmware

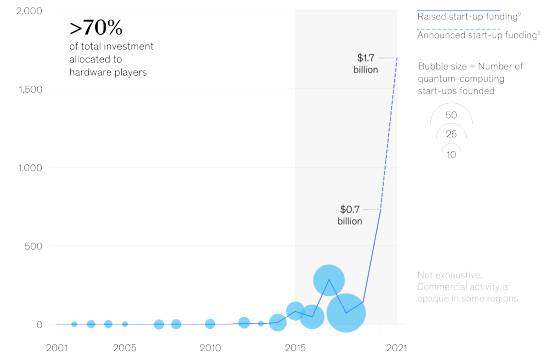
QINDOM
STRANGE WORKS
1QBit
Q-CTRL
Aliro QUANTUM
quantum benchmark
CQC
Labber QUANTUM
QUANTUM MACHINES
QILIMANJARO

Quantum Hardware

IONQ
PsiQ
ColdQuanta
EeroQ
IQM
QUANTUM FACTORY
QUANDELA
Quantum Microwave

Start-up activity and investments in quantum computing have skyrocketed since 2015.

Volume[1] of raised funding, $ millions



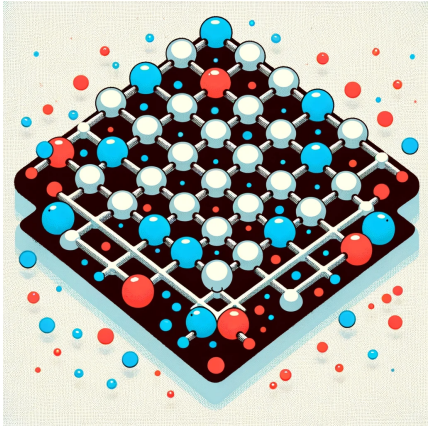
[1]Based on public investment data recorded in PitchBook; actual investment is likely higher.
[2]Public announcements of major deals; actual investment is likely higher.
[3]Start-ups from 2019 and later are likely still in stealth mode or are not yet recognized as quantum-computing companies by relevant platforms and experts.
Source: PitchBook; McKinsey analysis

Quantum Systems

Compute quantum mechanics

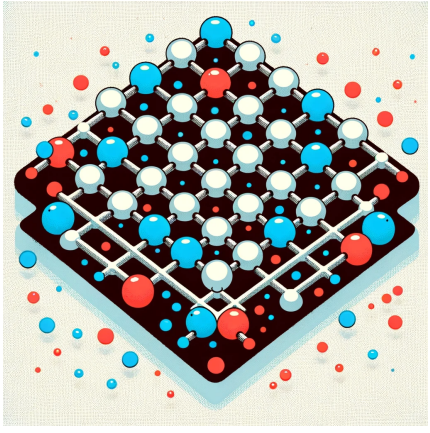
Compute quantum mechanics

- Representing N particles is difficult;



Compute quantum mechanics

- Representing N particles is difficult;
- considering N spins $(\uparrow, \downarrow)$, we deal with a 2^N dimensional Hilbert space!



What can we do?

What can we do?

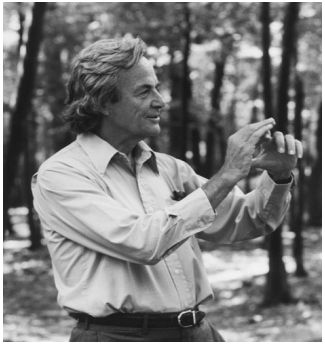
1. we can try to use classical methods to represent the system;

What can we do?

1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.

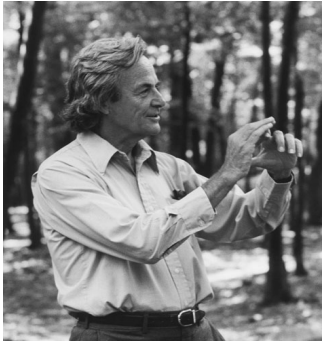
What can we do?

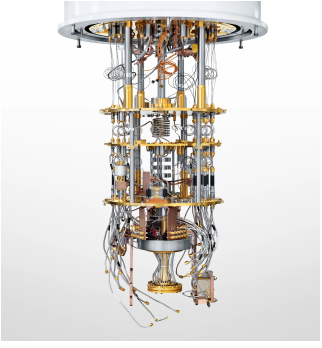
1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.



What can we do?

1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.



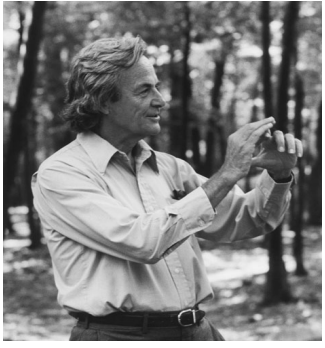


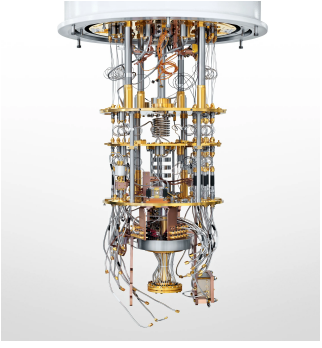
Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

Richard Feynman, 1982, Simulating Physics with Computers

What can we do?

1. **we can try to use classical methods to represent the system**;
2. we can build a quantum computer.





Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

Richard Feynman, 1982, Simulating Physics with Computers

Can we represent a state with a classical computer?

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires $\sim$ 1 billion of complex numbers;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires ~ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires $\sim$ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;
6. no problem. Let's get serious: `Fugaku`!

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

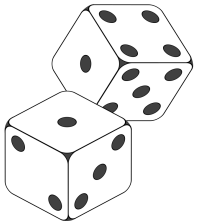
1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires $\sim$ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;
6. no problem. Let's get serious: `Fugaku`!

Some smart classical strategies

Some smart classical strategies

1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:

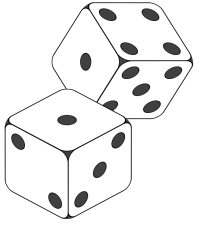
$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$



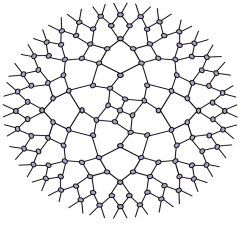
arXiv:1508.02989

Some smart classical strategies

1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:
$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$
2. Tensor Networks (TNs): contraction of complex systems into simpler structures;



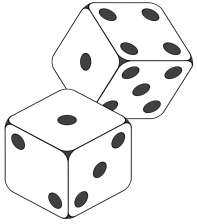
arXiv:1508.02989



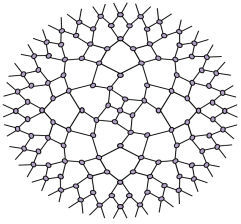
arXiv:1708.00006

Some smart classical strategies

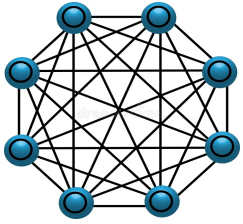
1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:
$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$
2. Tensor Networks (TNs): contraction of complex systems into simpler structures;
3. Neural Network Quantum States: use complex ANNs to represent the state.



arXiv:1508.02989



arXiv:1708.00006



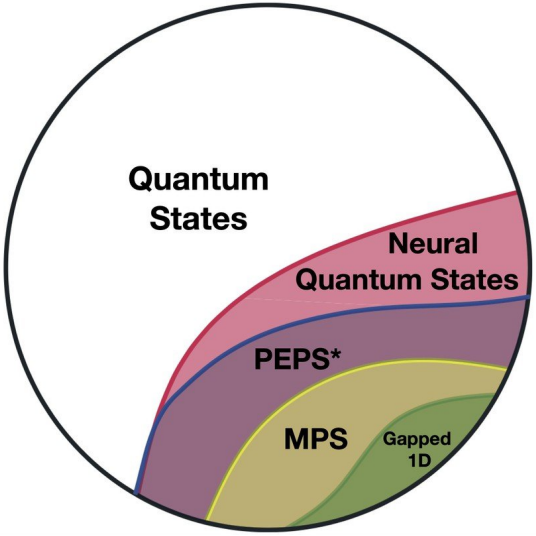
arXiv:1606.02318

Is this enough?

arXiv:2103.10293

Is this enough?

arXiv:2103.10293



Quantum computing

Qubits

Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

Superconducting loops
A resistance-free current oscillates back and forth around a circuit loop. An injected microwave signal excites the current into super-position states.

Trapped ions
Electrically charged atoms, or ions, have quantum energies that depend on the location of electrons. Tuned lasers cool and trap the ions, and put them in superposition states.

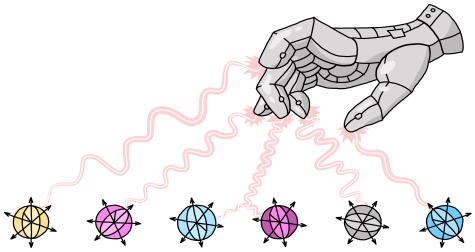
Silicon quantum dots
These "artificial atoms" are made by adding an electron to a small piece of pure silicon. Microwaves control the electron's quantum state.

Topological qubits
Quasiparticles can be seen in the behavior of electrons channeled through semi-conductor structures. Their braided paths can encode quantum information.

Diamond vacancies
A nitrogen atom and a vacancy add an electron to a diamond lattice. Its quantum spin state, along with those of nearby carbon nuclei, can be controlled with light.

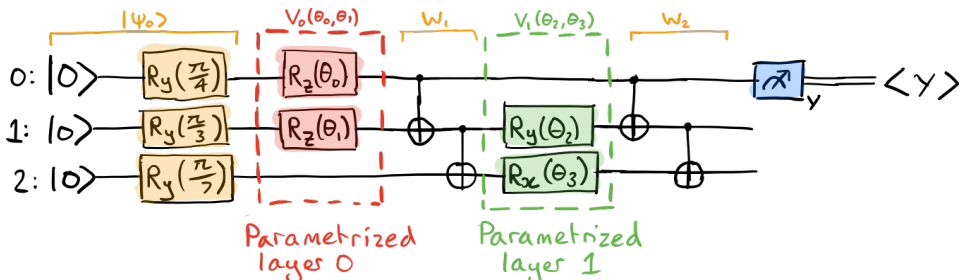
Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!



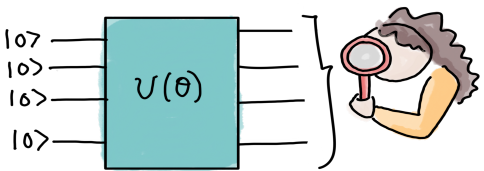
Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!
3. combine together gates to build **quantum circuits**;



Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!
3. combine together gates to build **quantum circuits**;
4. to access the information we need to measure the system.



What is a qubit?

What is a qubit?

Let us consider a two-dimensional Hilbert space, we define the computational basis:

$$|0\rangle \to \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad |1\rangle \to \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

A quantum bit (**qubit**) is the basic unit of quantum information and it is written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where α, β are complex numbers and the state is normalized, i.e. $|\alpha|^2 + |\beta|^2 = 1$.

Multiple qubits states

A system with n qubits lives in 2^n-dimensional Hilbert space, defining the basis:

$$|0\rangle_n = |00\ldots00\rangle\,,\ |1\rangle_n = |00\ldots01\rangle\,,\ |2\rangle_n = |00\ldots10\rangle\,,\ \ldots,\ |2^n-1\rangle_n = |11\ldots1\rangle$$

therefore a generic n qubits state is defined as

$$|\psi_n\rangle = \sum_{i=0}^{2^n-1} \alpha_i\,|i\rangle_n \quad \text{with} \quad \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$$

i.e. a superposition state vector with 2^n **complex numbers**.

Quantum circuits

The quantum circuit model considers a sequence of unitary quantum gates:

$$|\psi'\rangle = U_2 U_1 |\psi\rangle \quad \rightarrow \quad |\psi\rangle \text{ —} \boxed{U_1} \text{—} \boxed{U_2} \text{—} |\psi'\rangle$$

The final state $|\psi'\rangle$ is given by:

$$\psi'(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma'}} U_1 U_2(\boldsymbol{\sigma}, \boldsymbol{\sigma'}) \psi(\sigma_1, \ldots \sigma'_{i_1}, \ldots, \sigma'_{i_{N_{\text{targets}}}}, \ldots, \sigma_N),$$

where the sum runs over qubits targeted by the gate.

- U_2 and U_1 are gate matrices which act on the state vector.
- ψ is a state and it is bounded by memory.

Quantum gates

Single-qubit gates

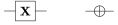
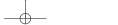
- Pauli gates
- Hadamard gate
- Phase shift gate
- Rotation gates

Two-qubit gates

- Controlled gates
- Swap gate
- fSim gate

Three-qubit gates

- Toffoli

Operator	Gate(s)	Matrix
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Pauli gates

X gate

The X gate acts like the classical NOT gate, it is represented by the σ_x matrix,

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

therefore

$$|0\rangle \text{ —}\boxed{X}\text{— } |1\rangle$$

$$|1\rangle \text{ —}\boxed{X}\text{— } |0\rangle$$

Z gate

The Z gate flips the sign of $|1\rangle$, it is represented by the σ_z matrix,

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

therefore

$$|0\rangle \text{ —}\boxed{Z}\text{— } |0\rangle$$

$$|1\rangle \text{ —}\boxed{Z}\text{— } -|1\rangle$$

Hadamard gate

The Hadamard gate (H gate) is defined as

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Therefore it creates a superposition of states

$$|0\rangle \text{ —}\boxed{H}\text{— } \frac{|0\rangle + |1\rangle}{\sqrt{2}} \equiv |+\rangle$$

$$|1\rangle \text{ —}\boxed{H}\text{— } \frac{|0\rangle - |1\rangle}{\sqrt{2}} \equiv |-\rangle$$

Measurements

In real experiments we perform measurements with a preselected number of shots.

Shots contribute to the reconstruction of the underlying wave-function distribution.

Measurement (M) gate:

Lets consider the following circuit:

The analytic final state is:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

When measuring the final state we obtain 0 or 1 each with 50% probability.

New computational power: an example

New computational power: an example

With quantum computing, we introduce new tools.

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ |1\rangle = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) =$$

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|00\rangle + \text{NOT}_{\text{targ}}|10\rangle) =$$

New computational power: an example

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \; |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\Big(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\Big) = \frac{1}{\sqrt{2}}(|00\rangle + \text{NOT}_{\text{targ}}|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

New computational power: an example

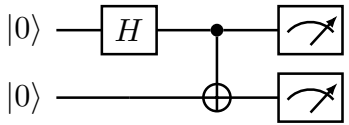
With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\Big(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\Big) = \frac{1}{\sqrt{2}}(|00\rangle + \text{NOT}_{\text{targ}}|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$



Parametric gates prepare variational quantum states

Parametric gates prepare variational quantum states

- Among the gates, parametric ones can be useful!

Parametric gates prepare variational quantum states

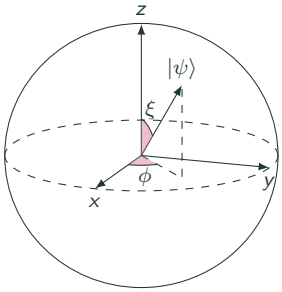
- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Parametric gates prepare variational quantum states

- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

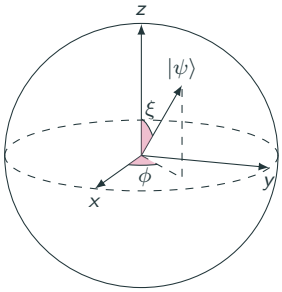
$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with} \qquad \alpha = \cos\frac{\theta}{2}, \quad \beta = e^{i\phi}\sin\frac{\theta}{2}.$$

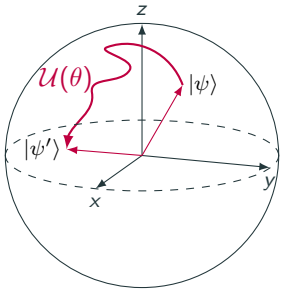


Parametric gates prepare variational quantum states

- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with} \qquad \alpha = \cos\frac{\theta}{2}, \quad \beta = e^{i\phi}\sin\frac{\theta}{2}.$$





We can use as parametric gates the rotation around the axis of the block sphere:

$$R_k(\theta) = \exp\left[-i\theta\sigma_k\right], \qquad \text{with} \qquad \sigma_k \in \{I, \sigma_x, \sigma_y, \sigma_z\}.$$

But can this work?

But can this work?

1. Depend on the problem

But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!

But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

But can this work?

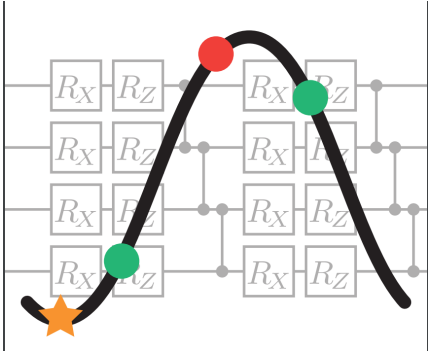
1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

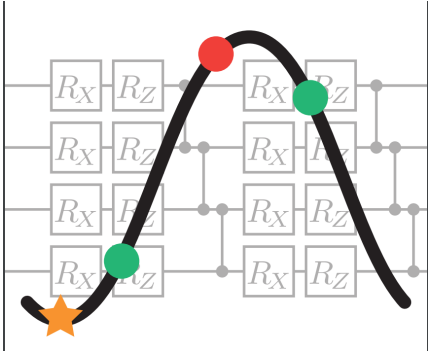


But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;

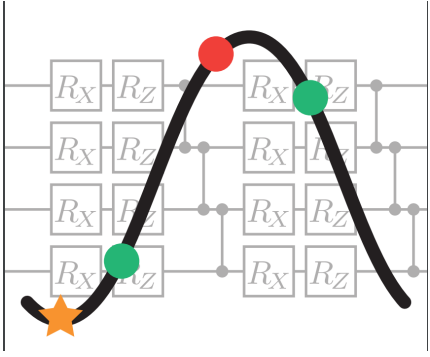


But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;

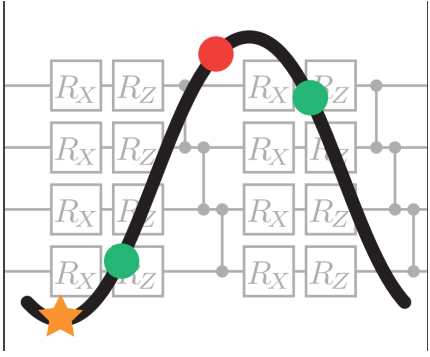


But can this work?

1. Depend on the problem, e.g. playing in the Hilbert space makes sense when tackling a many body problem!
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

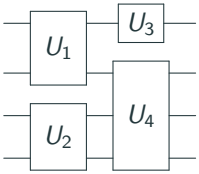
1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;
3. **Solovay-Kitaev theorem**: the number of gates needed by $\mathcal{U}$ to represent V with precision δ is $\mathcal{O}(\log^c \delta^{-1})$, where $c < 4$.



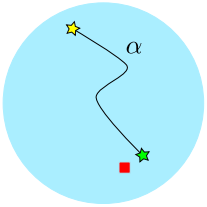
Rational for Variational Quantum Circuits

Rational: Deliver variational quantum states $\rightarrow$ explore a large Hilbert space.

$$U(\vec{\alpha}) = U_n \dots U_2 U_1$$



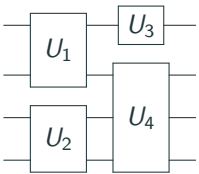
Near optimal solution



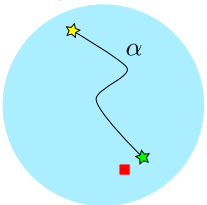
Rational for Variational Quantum Circuits

Rational: Deliver variational quantum states $\rightarrow$ explore a large Hilbert space.

$$U(\vec{\alpha}) = U_n \dots U_2 U_1$$



Near optimal solution



Idea: Quantum Computer is a machine that generates variational states.

$\Rightarrow$ **Variational Quantum Computer**

Solovay-Kitaev Theorem

Let $\{U_i\}$ be a dense set of unitaries.
Define a circuit approximation to V:

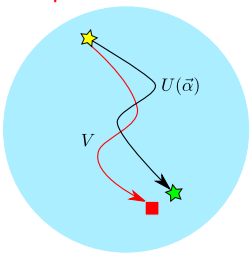
$$|U_k \dots U_2 U_1 - V| < \delta$$

Scaling to best approximation

$$k \sim \mathcal{O}\left(\log^c \frac{1}{\delta}\right)$$

where $c < 4$.

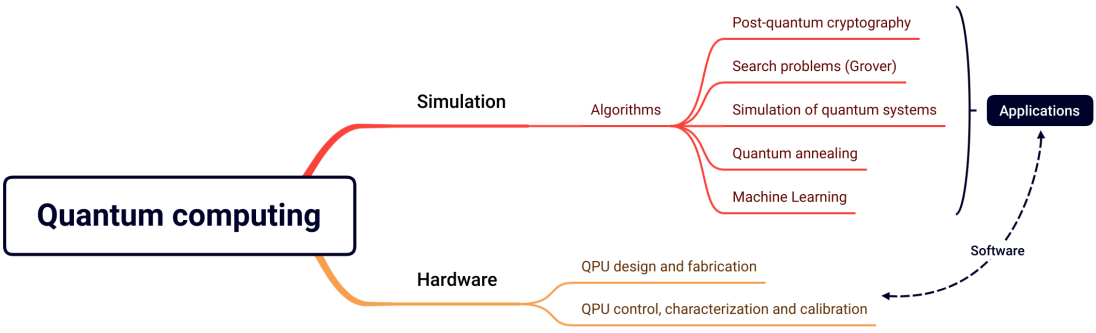
Optimal solution



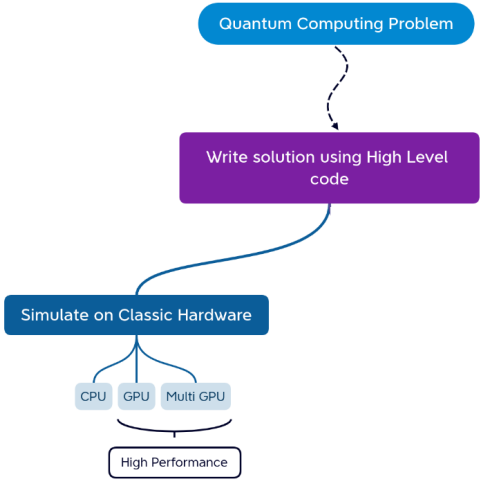
$\Rightarrow$ The approximation is efficient and requires a finite number of gates.

Quantum software challenges

Software and Quantum Computing



Full-stack deployment process



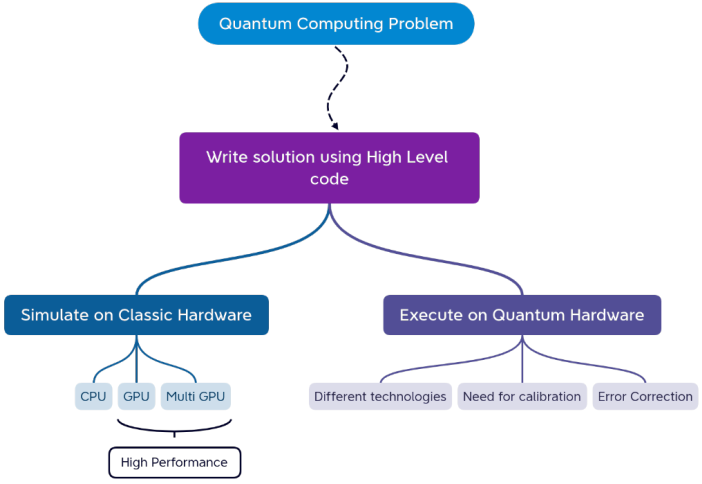
1. **Prototyping**

   High-level programming language

2. **Simulation**

   Fast classical quantum simulation

Full-stack deployment process



1. **Prototyping**

   High-level programming language

2. **Simulation**

   Fast classical quantum simulation

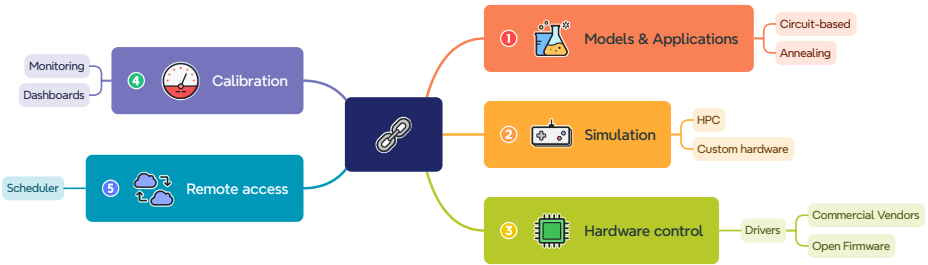
3. **Deployment on hardware**

   Hardware optimization

Software challenges

Requirements for self-hosted quantum hardware:

1. Access to **interdisciplinary set of software tools** for:



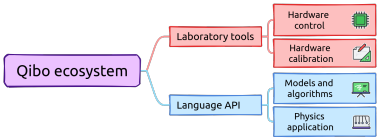
2. **Open-source software** tools supported by benchmarks and publications.

Introducing Qibo

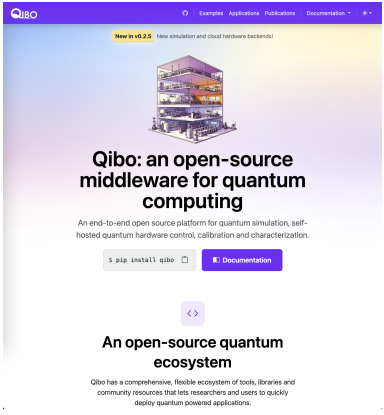
The Qibo ecosystem

arXiv:2009.01845

Qibo is an **open-source** hybrid quantum operating system for self-hosted quantum computers.



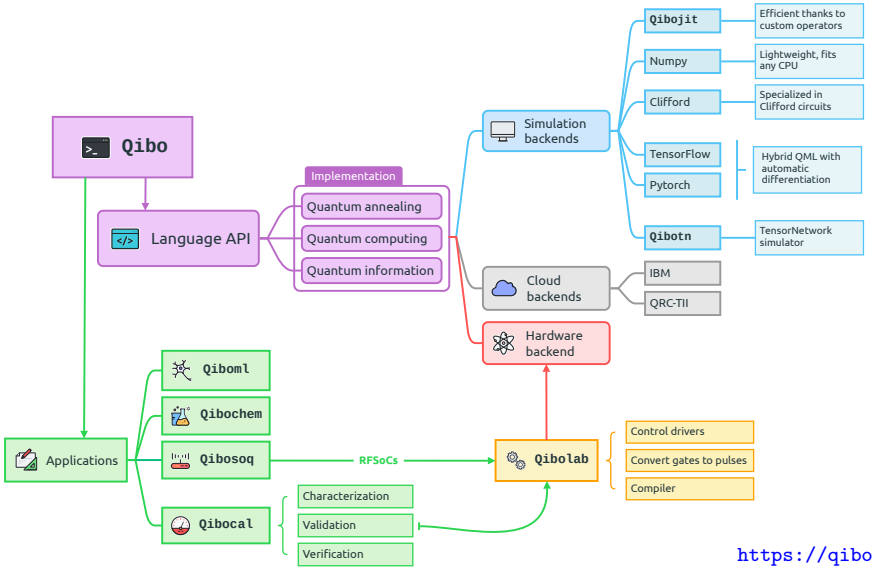
1. Fully open-source and community driven.
2. Modular layout design with possibility of adding:
   - new backends for simulation,
   - new platforms for hardware control,
   - new drivers for control electronics.
3. Supported by documentation and tests/CI on quantum hardware.



https://qibo.science

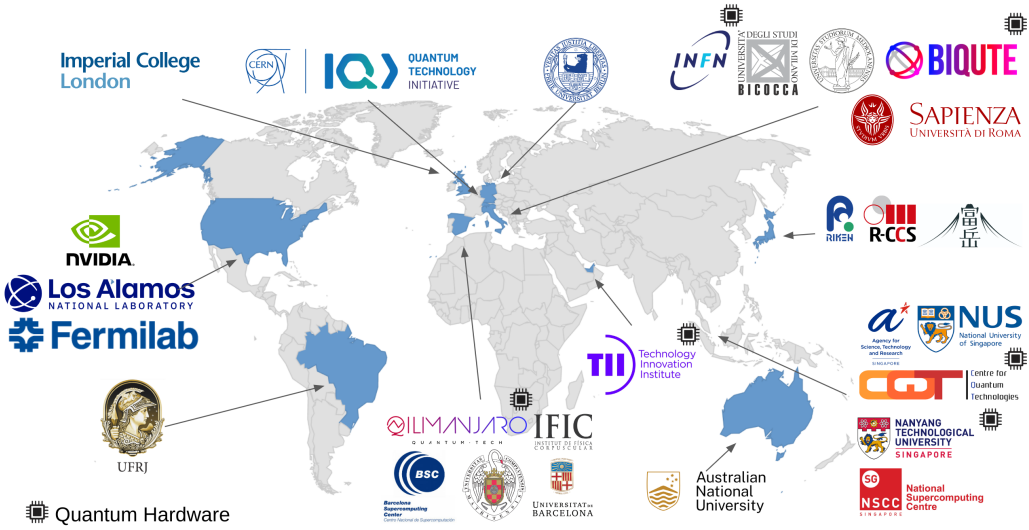
Qibo's macrostructure

arXiv:2009.01845



https://qibo.science

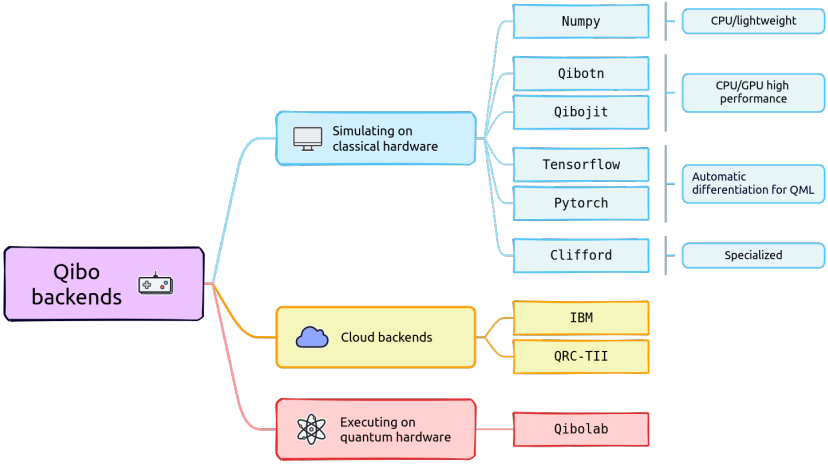
Qibo Contributors (March 2024)



Quantum Simulation

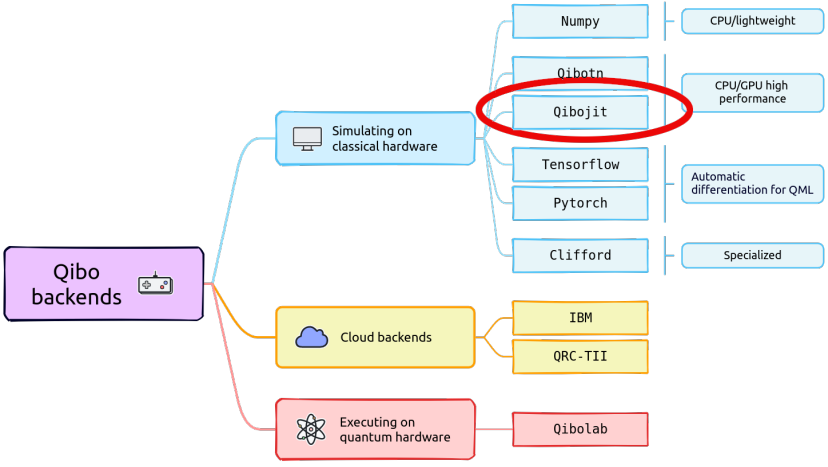
Qibo backends

We provide a modular and extensible framework for quantum computing based on **backends**:



Qibo backends

We provide a modular and extensible framework for quantum computing based on **backends**:

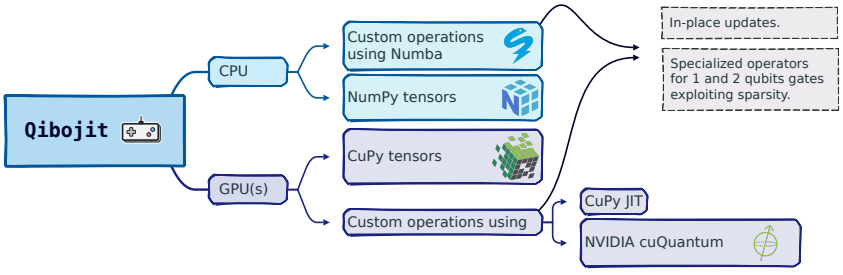


State vector simulation solves:

$$\psi'(\sigma_1, \ldots, \sigma_n) = \sum_{\tau'} G(\tau, \tau')\psi(\sigma_1, \ldots, \tau', \ldots, \sigma_n)$$

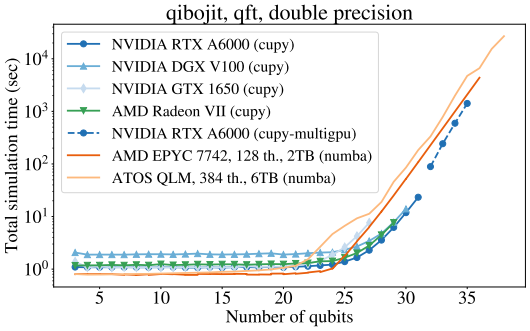
The number of operations scales exponentially with the number of qubits.

Qibo uses just-in-time technology and hardware acceleration:



Classical quantum simulation benchmarks

arXiv:2203.08826

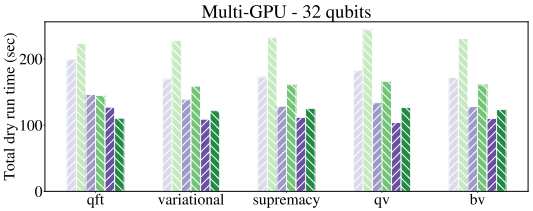


Major features:

Supports CPU, GPU and multi-GPU.

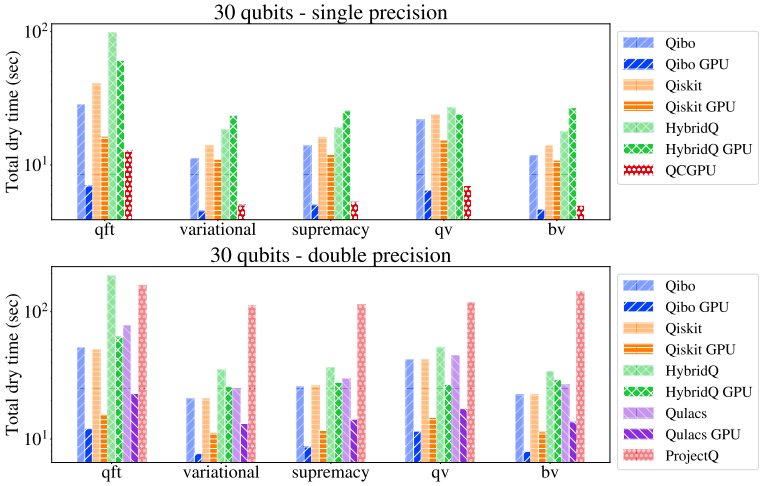
NVIDIA and AMD GPUs support.

Reduced memory footprint.



Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks`

Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks`

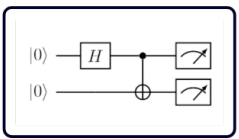


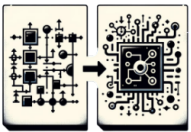
Laboratory modules

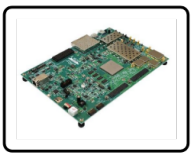
How to control qubits?

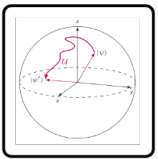
arXiv:2308.06313

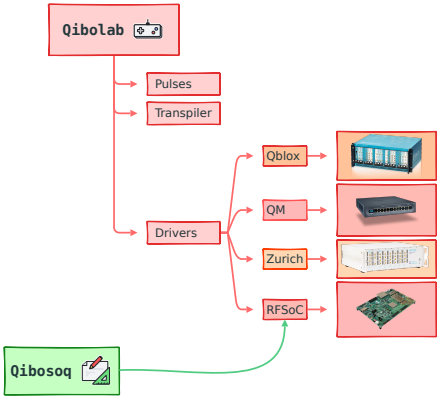
Circuit definition → Circuit transpilation → Hardware deployment → Experiment execution











Major features:

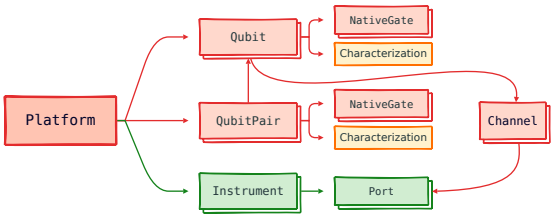
Pulse and pulse sequence API.

Extensible API to drivers of control instruments.

Hardware sweeps for faster execution of calibration routines.

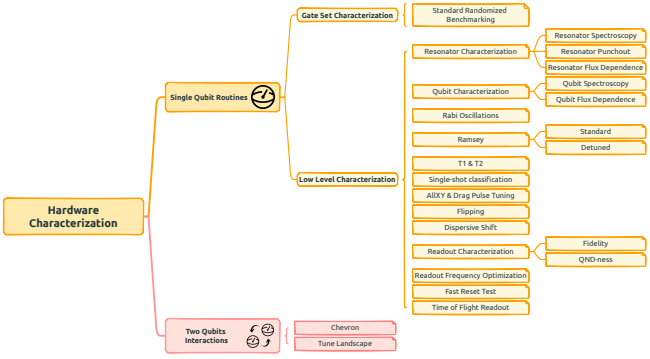
Transpilers from arbitrary circuits to pulses.

`C/C++` and `Rust` APIs



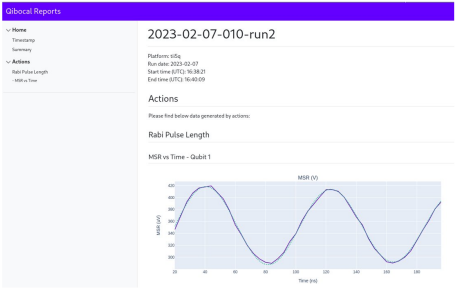
Qubit characterization and calibration with Qibocal

arXiv:2303.10397



Major features:

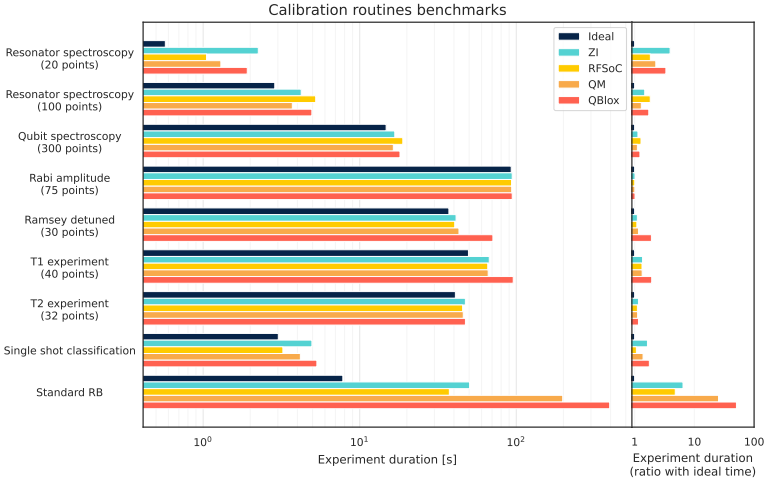
Calibration of single and multi-qubit platforms are possible using **Qibo**.



Benchmarking instruments performance

arXiv:2308.06313

We compare the ideal pulse sequence execution performance to instruments execution duration.



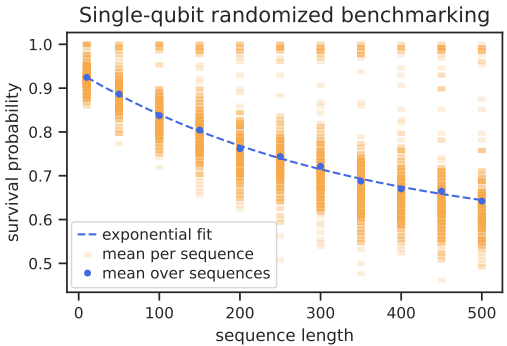
Zurich Instruments (ZI), Quantum Machines (QM), QBlox and RFSoC FPGA (Qibosoq+QICK).

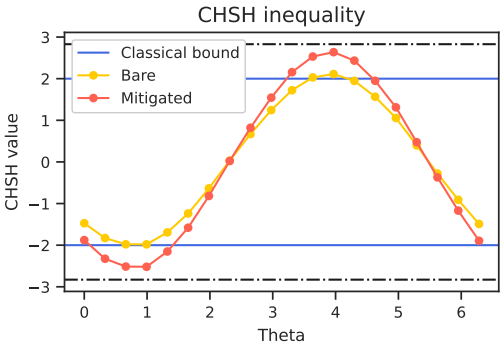
Full-stack applications

Full-stack deployment: RB and Bell inequality

arXiv:2308.06313

Examples of results executed on quantum hardware (single-qubit) after calibration with Qibo:

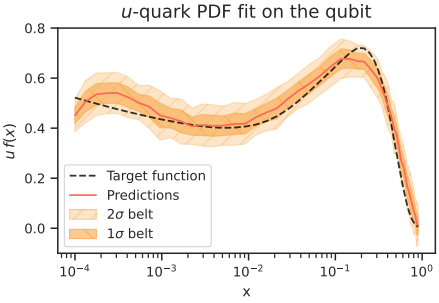




Full-stack procedure: data regression

arXiv:2308.06313

1. **High-Level API (Qibo)**
   - Define model prototype.
   - Implement training loop.
   - Perform training using simulation.
2. **Calibration (Qibocal)**
   - Calibrate qubit.
   - Generate platform configuration.
3. **Execution (Qibolab)**
   - Allocate calibrated platform.
   - Compile and transpile circuit.
   - Execute model and return results.



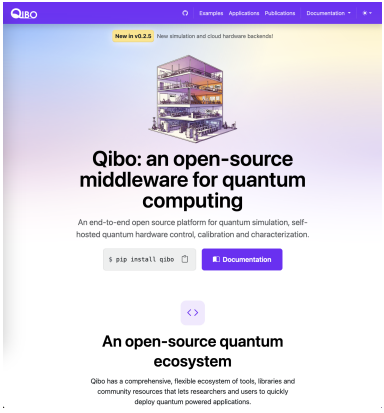
Parameter	Value
N_{data}	50 points
N_{shots}	500
MSE	10^{-3}
Electronics	Xilinx ZCU216
Training time	< 1h

Outlook

What next?

Hands-on tutorials:

1. Setting up Qibo's quantum middleware software environment, building quantum circuits.
2. Quantum circuits, gates and shots. Bell states and ancillary qubits.
3. Grover search algorithm.
4. Introducing quantum noise and quantum hardware limitations
5. Mitigating quantum noise via Quantum Error Mitigation



https://qibo.science