

# Introducing Qibo

Towards a hybrid quantum operating system

---

Stefano Carrazza

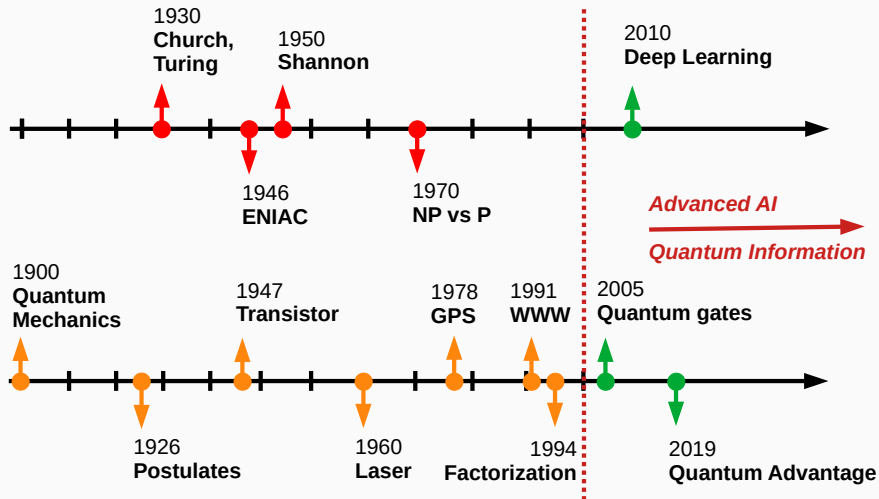
July 9, 2022

ICHEP2022, Bologna

# Introduction

---

# The Quantum Disruption



# Towards quantum computing

From a practical point of view, we are moving towards new technologies, in particular **hardware accelerators**:

**CPU**



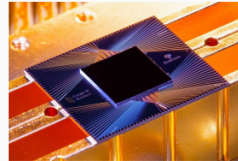
**GPU**



**FPGA/ASIC**



**Quantum chip**

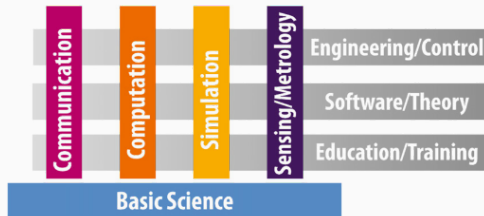


Moving from **general purpose devices**  $\Rightarrow$  **application specific**



# 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.

⇒ Entanglement, superposition, interference, etc.

# Qubits

---

# What is a qubit?

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

$$|0\rangle \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

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

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \rightarrow \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

where  $\alpha, \beta$  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:

$$|0\rangle_n = |00 \dots 00\rangle, |1\rangle_n = |00 \dots 01\rangle, |2\rangle_n = |00 \dots 10\rangle, \dots, |2^n - 1\rangle_n = |11 \dots 1\rangle$$

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:

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

where the sum runs over qubits targeted by the gate.

- $U_2$  and  $U_1$  are gate matrices which act on the state vector.
- $\psi$  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  $\sigma_x$  matrix,

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

therefore

$$\begin{aligned} |0\rangle &\xrightarrow{X} |1\rangle \\ |1\rangle &\xrightarrow{X} |0\rangle \end{aligned}$$

## $Z$ gate

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

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

therefore

$$\begin{aligned} |0\rangle &\xrightarrow{Z} |0\rangle \\ |1\rangle &\xrightarrow{Z} -|1\rangle \end{aligned}$$

# Hadamard gate

The Hadamard gate ( $H$  gate) is defined as

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

Therefore it creates a superposition of states

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

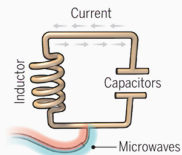
$$|1\rangle \xrightarrow{H} \frac{|0\rangle - |1\rangle}{\sqrt{2}} \equiv |-\rangle$$



# Quantum technology

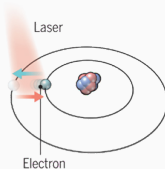
---

# Quantum Technologies



## Superconducting loops

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



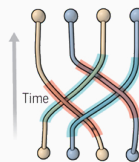
## Trapped ions

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.



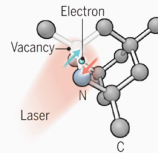
## Silicon quantum dots

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 semiconductor 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.

### Number entangled

9

14

2

N/A

6

### Company support

Google, IBM, Quantum Circuits

ionQ

Intel

Microsoft, Bell Labs

Quantum Diamond Technologies

### Pros

Fast working. Build on existing semiconductor industry.

Very stable. Highest achieved gate fidelities.

Stable. Build on existing semiconductor industry.

Greatly reduce errors.

Can operate at room temperature.

### Cons

Collapse easily and must be kept cold.

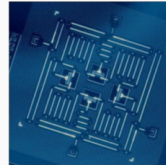
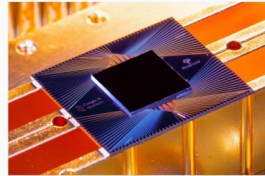
Slow operation. Many lasers are needed.

Only a few entangled. Must be kept cold.

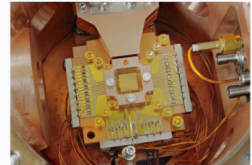
Existence not yet confirmed.

Difficult to entangle.

# Physical implementation



(a) Superconducting device assembled by IBM



(b) Chip based on trapped ions technology

# Quantum Algorithms

There are three families of algorithms:

## Gate Circuits

- Search (Grover)
- QFT (Shor)
- Deutsch
- ...

## Variational (AI inspired)

- Eigensolvers
- Autoencoders
- Classifiers
- ...

## Annealing

- Direct Annealing
- Adiabatic Evolution
- QAOA
- ...

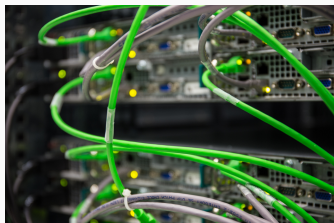
## Challenges

---

# Challenges

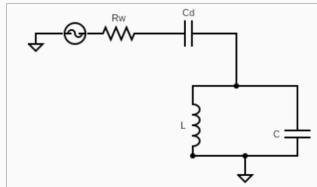
## Quantum tech. involves multiple challenges:

- simulate efficiently algorithms on classical hardware for QPU?
- control, send and retrieve results from the QPU?
- error mitigation, keep noise and decoherence under control?

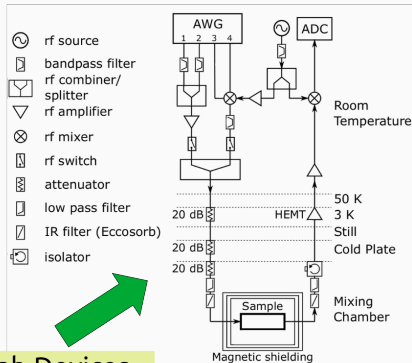


# Physical implementation

## Transmon (qubit)



## Platform design



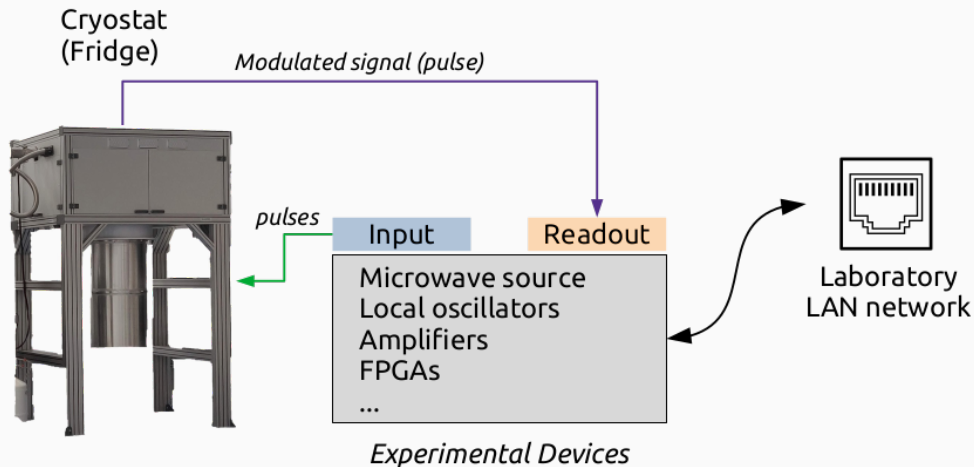
Lab Devices

## Cryostat (Fridge)



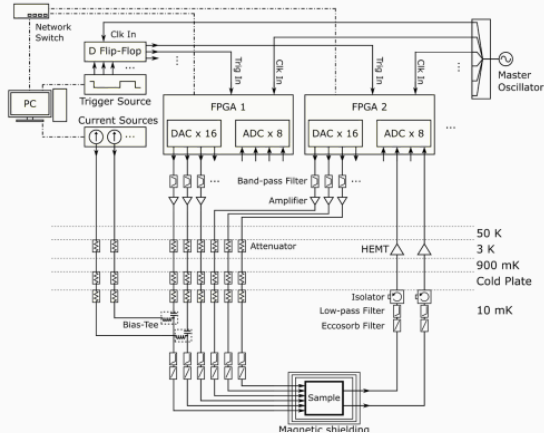
QRC@TII Labs

# Physical implementation





# Physical implementation

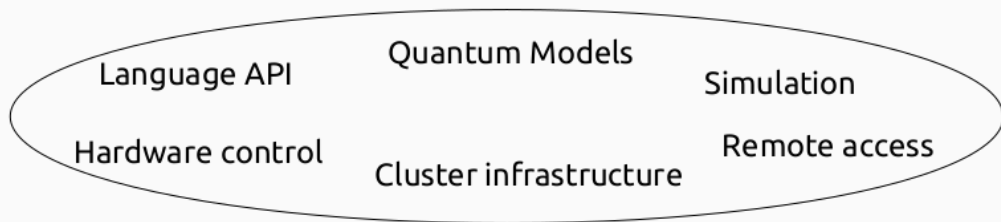


Platform **complexity increases** with qubits.

More **devices** are required.

Operation **scheduling** is necessary.

**Calibration** procedures must be periodic.

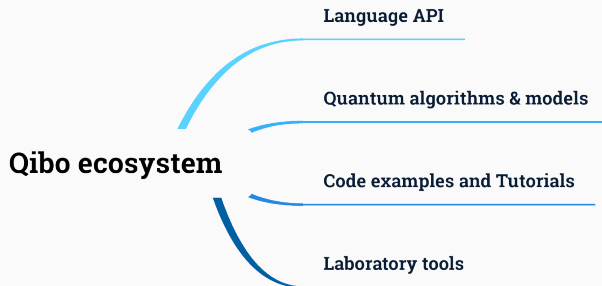


## Introducing Qibo

---

# Introducing Qibo

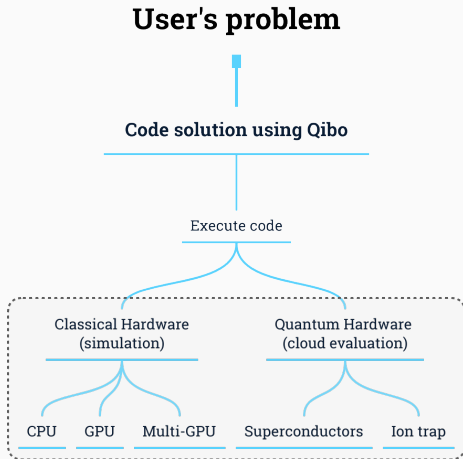
**Qibo** is an open-source full stack **API** for quantum simulation and hardware control. It is platform **agnostic** and supports **multiple backends**.



<https://github.com/qiboteam/qibo>

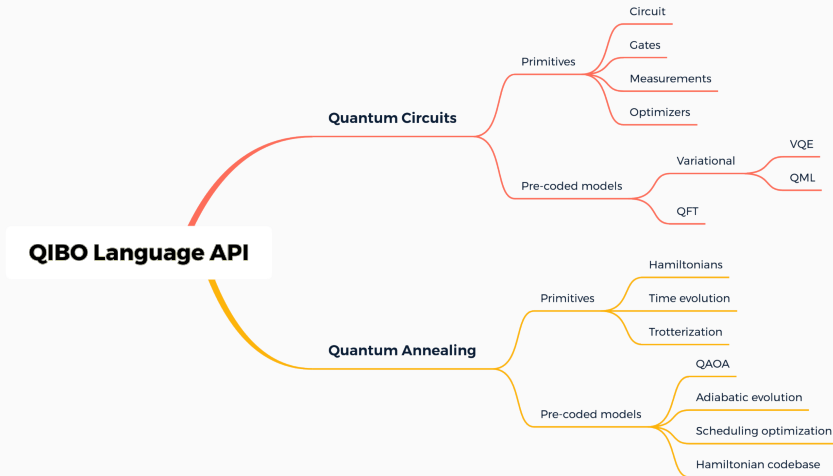
<https://arxiv.org/abs/2009.01845>



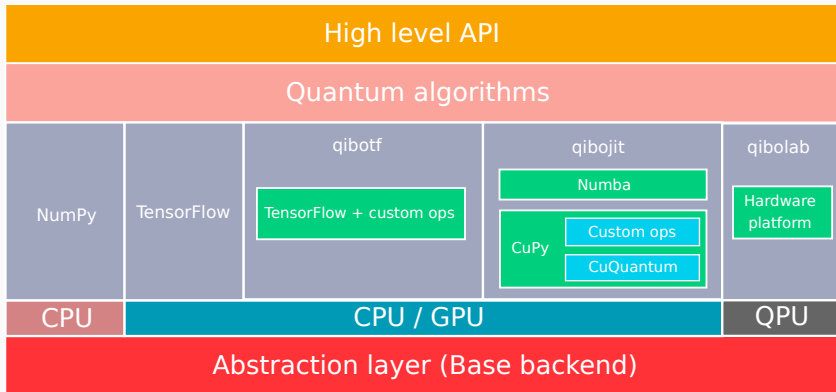


- Single piece of code
- Automatic deployment on simulators and quantum devices
- Plugin backends mechanism

# Computational models in Qibo



## Qibo stack





## **Qibo simulation benchmarks**

---

**qibojit** uses just-in-time technology for state vector simulation:

- on CPU → **Numpy** tensors and custom operations compiled with **Numba JIT**.

```
from numba import njit, prange

@njit(parallel=True, cache=True)
def apply_gate_kernel(state, gate, target):
    """Operator that applies an arbitrary one-qubit gate.

    Args:
        state (np.ndarray): State vector of size (2 ** nqubits,).
        gate (np.ndarray): Gate matrix of size (2, 2).
        target (int): Index of the target qubit.
    """
    k = 1 << target
    # for one target qubit: loop over half states
    nstates = len(state) // 2
    for g in prange(nstates):
        # generate index with fast binary operations
        i1 = ((g >> m) << (m + 1)) + (g & (k - 1))
        i2 = i1 + k
        state[i1], state[i2] = (gate[0, 0] * state[i1] + \
                                gate[0, 1] * state[i2],
                                gate[1, 0] * state[i1] + \
                                gate[1, 1] * state[i2])

    return state
```

**qibojit** uses just-in-time technology for state vector simulation:

- on CPU → **Numpy** tensors and custom operations compiled with **Numba JIT**.
- on GPU → **CuPy** tensors and custom operations with:
  - **CuPy** JIT Raw Kernels
  - **cuQuantum** Python API

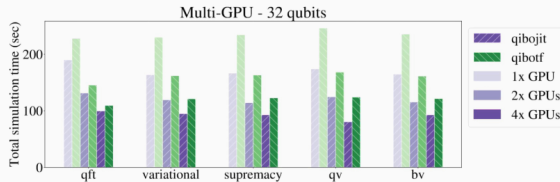
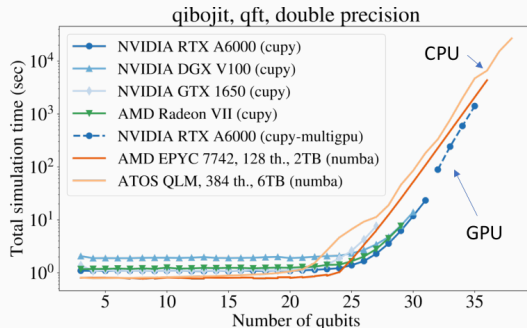
## Framework Integrations

cuQuantum is integrated with leading quantum circuit simulation frameworks. Download cuQuantum and get dramatically accelerated performance from your framework of choice, with zero code changes.



PENNY LANE





**Qibo** implements a high performance state vector simulation framework.

→ Supports CPU, GPUs and multi-GPU.

→ NVIDIA and AMD (ROCm) GPUs.

→ Reduced memory footprint.

```
import numpy as np
from qibo import models, gates

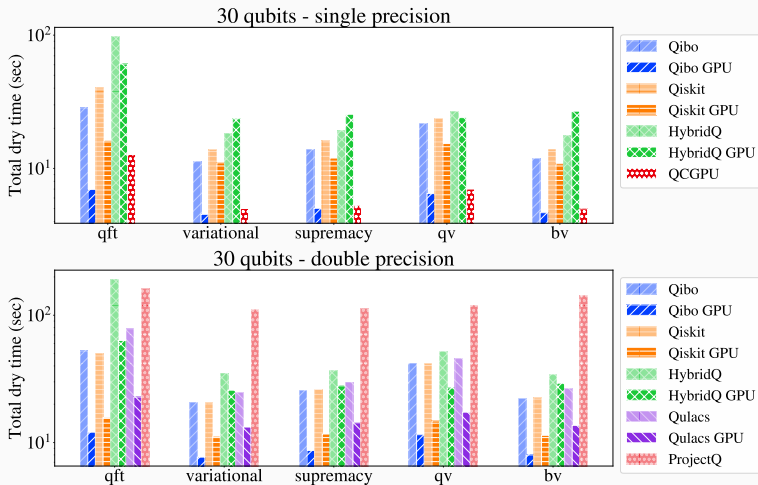
# create a circuit for N=3 qubits
circuit = models.Circuit(3)

# add some gates in the circuit
circuit.add([gates.H(0), gates.X(1)])
circuit.add(gates.RX(0, theta=np.pi/6))

# execute the circuit and obtain the
# final state as a tf.Tensor
final_state = circuit()
```

# Qibo vs other libraries

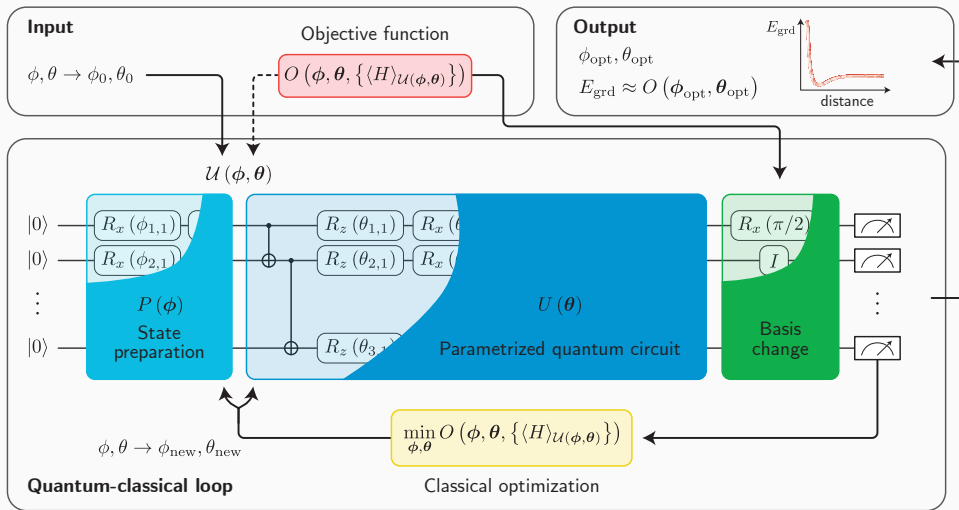
Benchmark library: <https://github.com/qiboteam/qibojit-benchmarks>



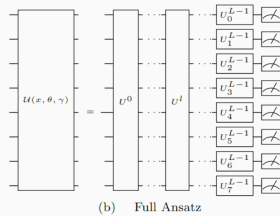
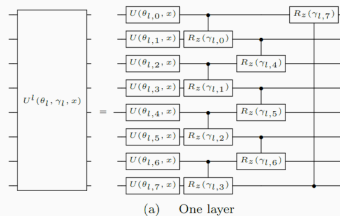
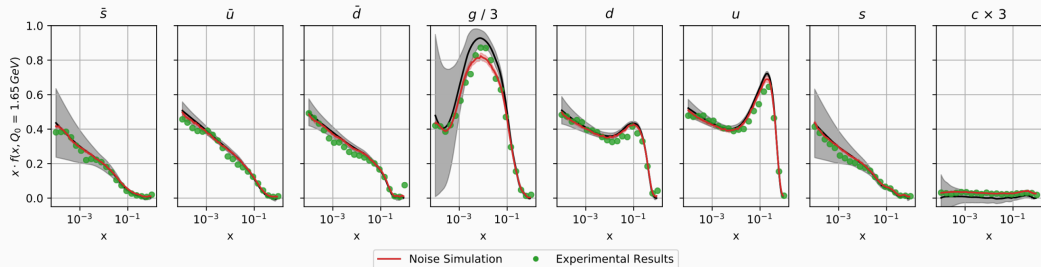
# Investigating models for HEP

---

# Novel quantum models for HEP



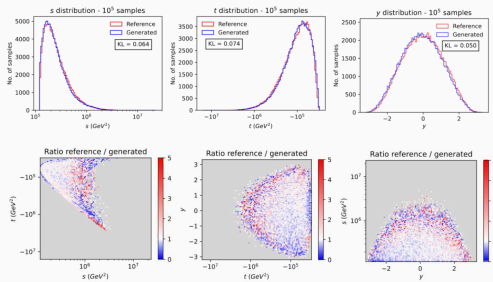
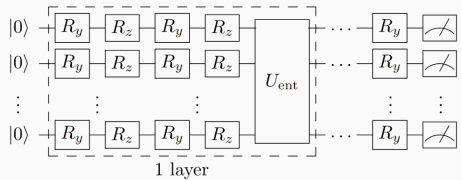
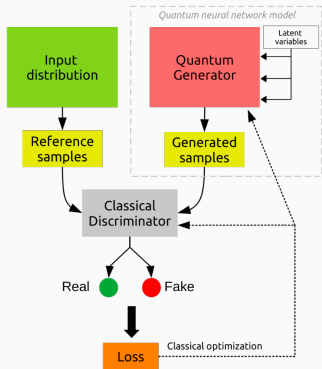
# Determination of parton distribution functions using QML



<https://arxiv.org/abs/2011.13934>



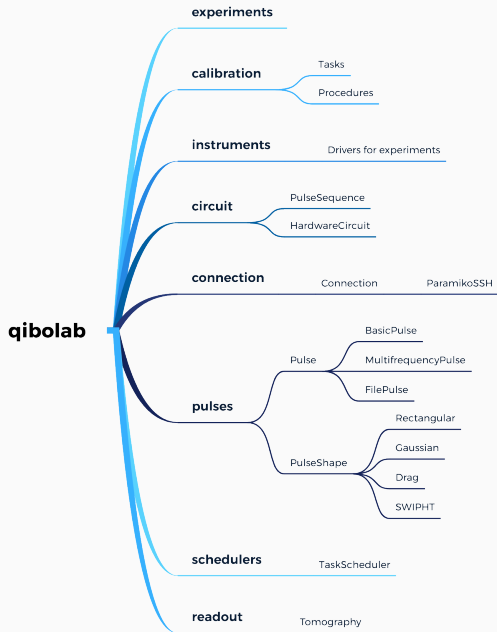
# MC event generation using Style-qGAN



<https://arxiv.org/abs/2110.06933>

# Quantum hardware control

---

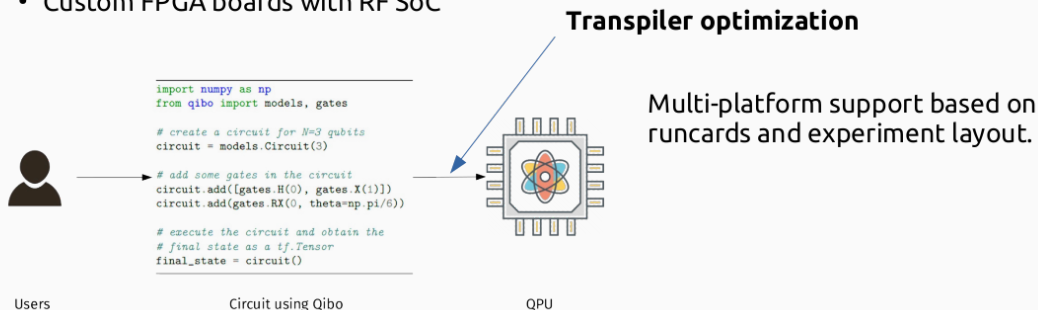


## QPU support using qibolab:

- Agnostic layout.
- Multiple experiments support.
- Plug & play for instruments.
- Tools for hardware control.

**Qibo** includes circuit to quantum hardware transpiler for:

- Wavefunction generators
- Local oscillators
- QBlox devices
- Custom FPGA boards with RF SoC



# Example

```
import qibo
from qibo import models, gates
import numpy as np

# select experimental backend
qibo.set_backend("qibolab", platform='tiiq')

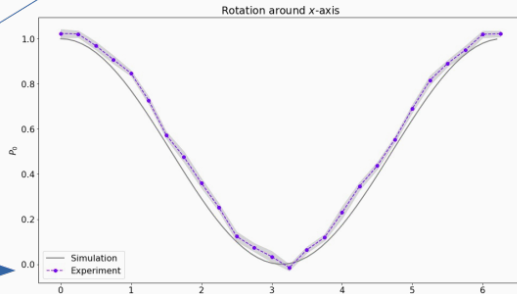
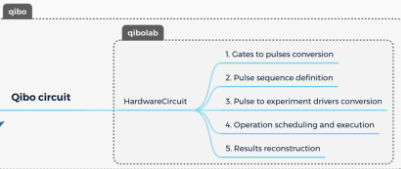
# create circuit
circuit = models.Circuit(1)
circuit.add(gates.RX(0, theta=0))
circuit.add(gates.M(0))

exp_angles = np.arange(0, 2*np.pi, 0.25)
for t in exp_angles:
    # set RX rotation angle
    circuit.set_parameters([t])

    # execute the circuit
    state = circuit.execute(nshots=1024)

    # calculate the probabilities of  $|0\rangle$  and  $|1\rangle$ 
    probs_0, probs_1 = state.probabilities(qubits=(0,))
```

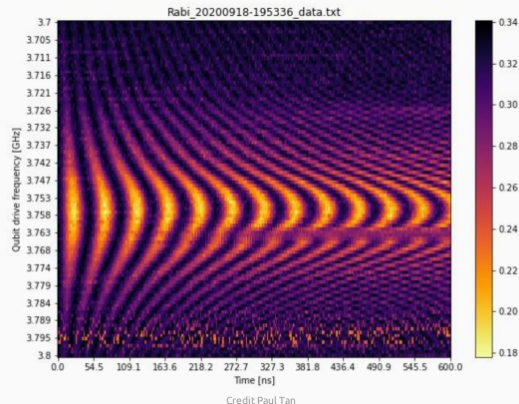
Lab User



# Example

**Qibo** includes qubit characterization and calibration procedures such as:

- Resonator spectroscopy
- Qubit spectroscopy
- Rabi pulse length, gain and amplitude
- Ramsey interferometry
- T1 determination
- Spin echo



# Outlook

---

**Qibo** is a framework for quantum research acceleration:

- Publicly available as an open-source project
- Designed with modular layout
- Community driven effort

We invite you to try out **Qibo**!

Code : <https://github.com/qiboteam/qibo>

Docs : <https://qibo.readthedocs.io>

