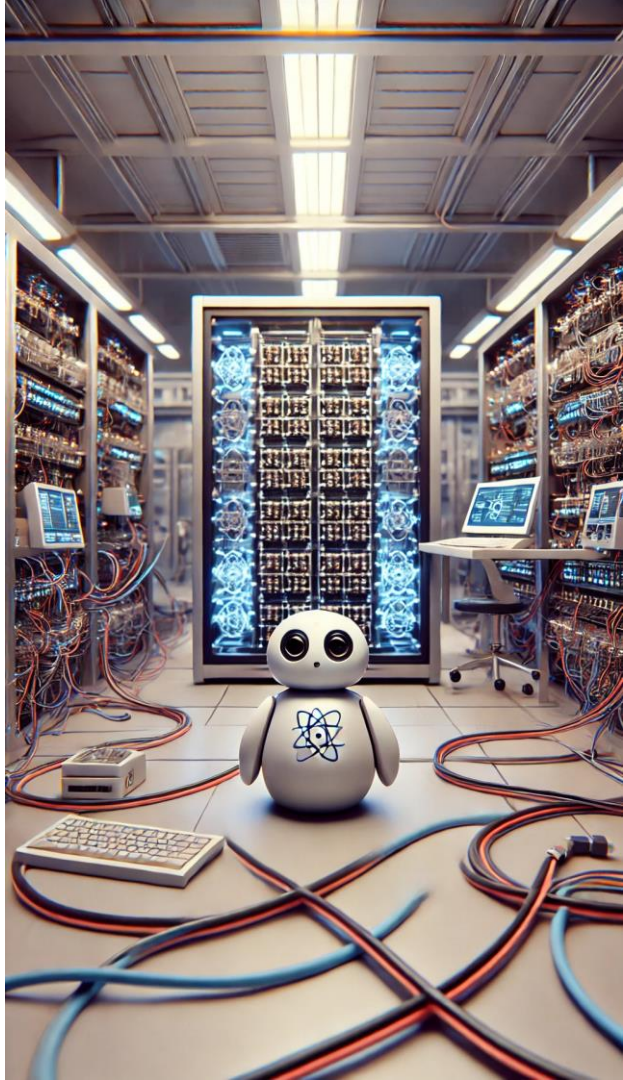
QIBO: An open-source modular framework for quantum computing

Speaker: Simone Bordoni
Project coordinator: Stefano Carrazza

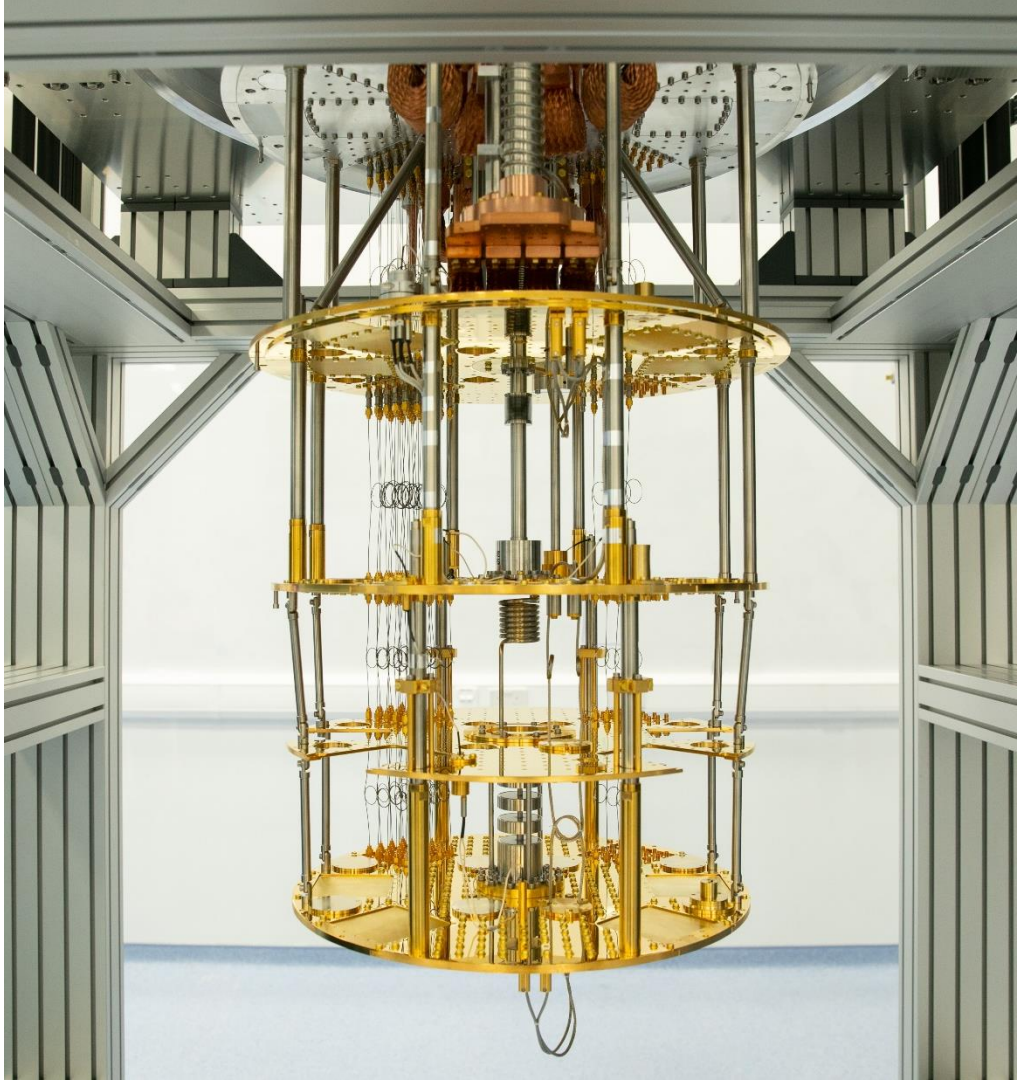
Qibo

QiboLab

QiboCal

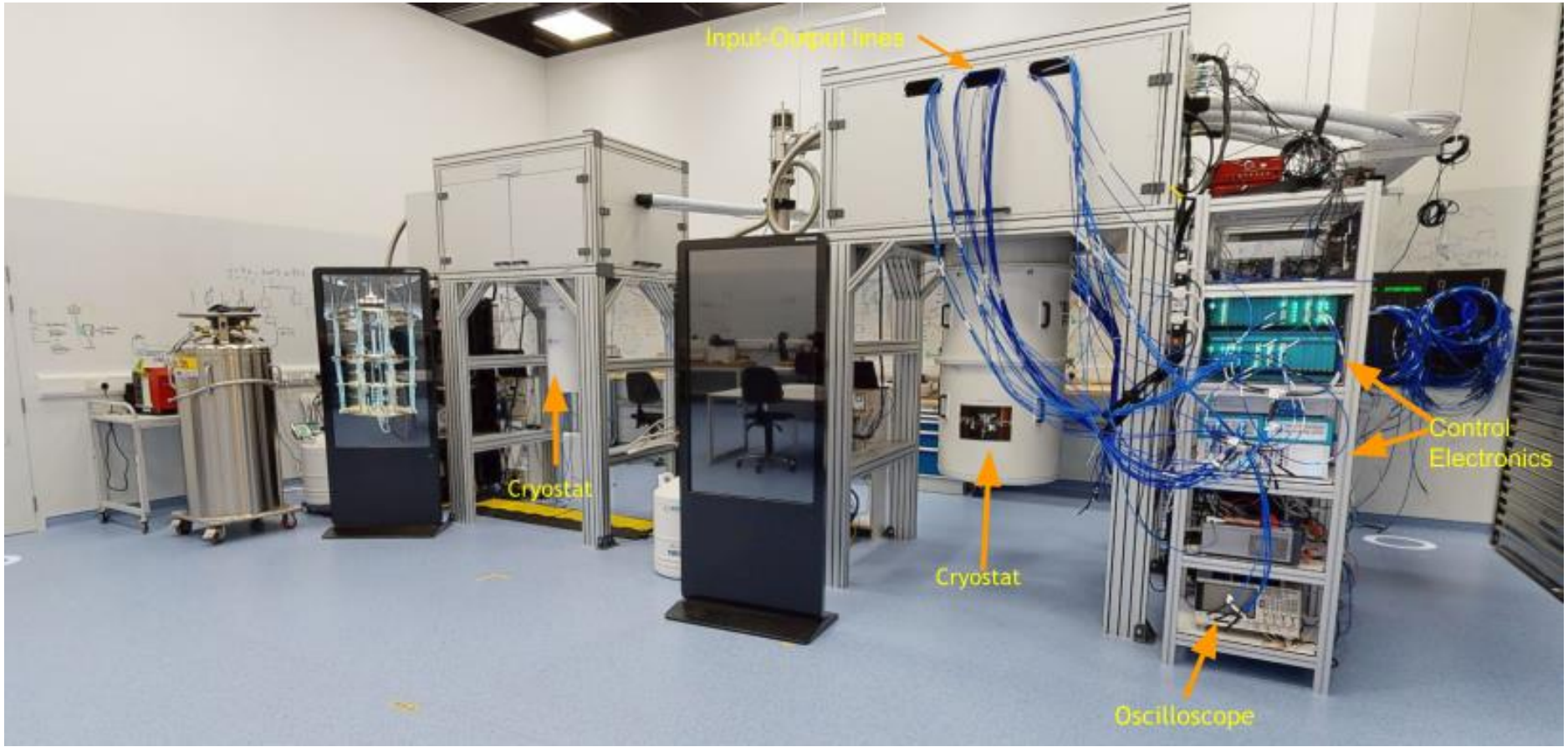


Introduction
Table of contents

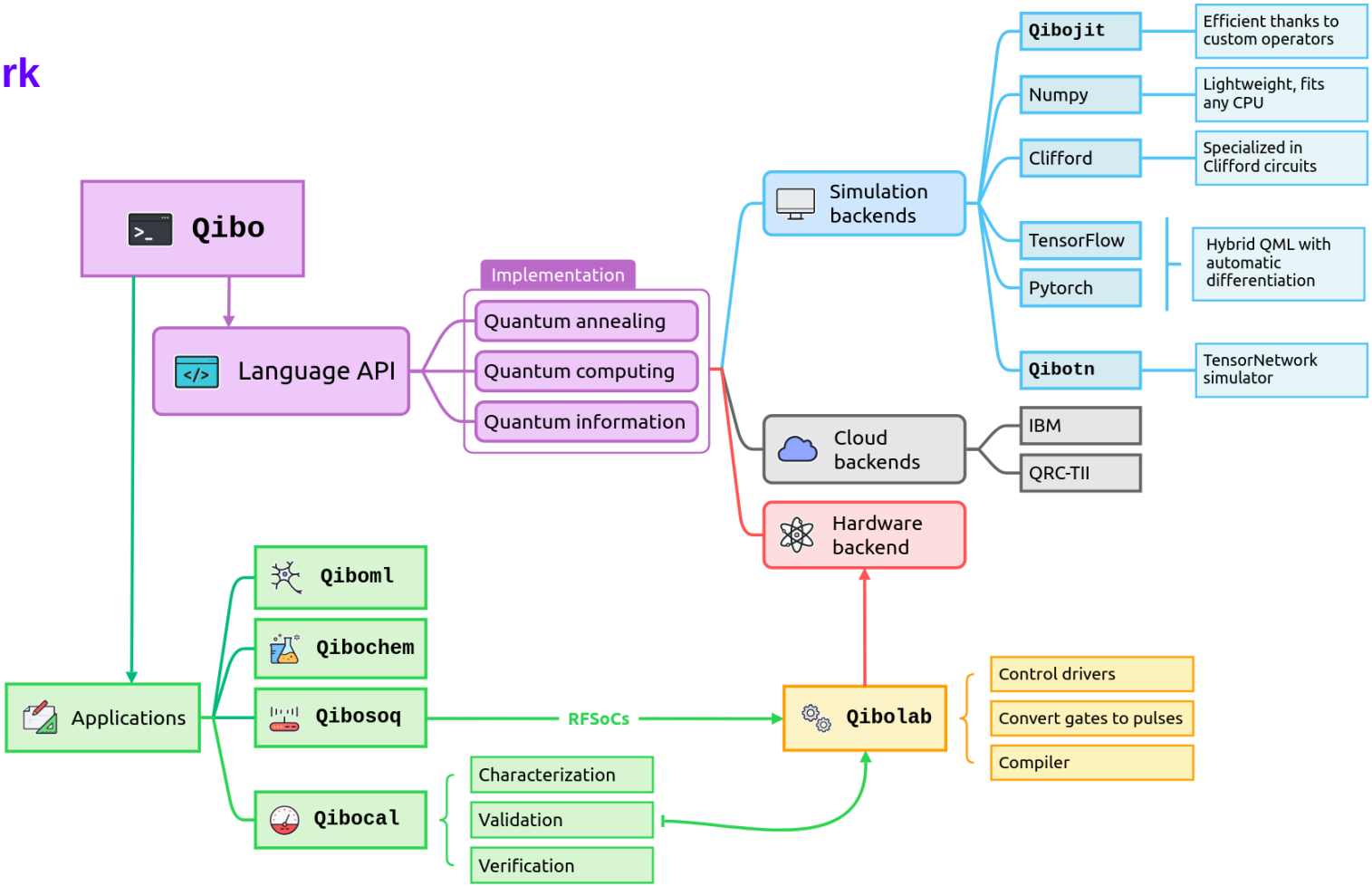


Introduction
Developers

Introduction
Lab



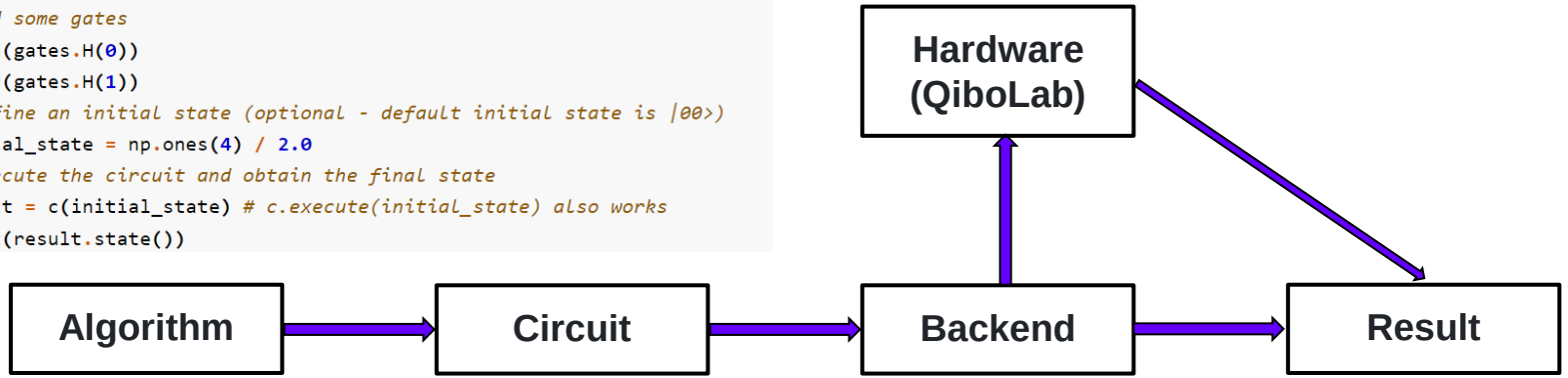
Qibo
Framework



Qibo
Circuit simulation

```python
import numpy as np
from qibo import Circuit, gates

# Construct the circuit
c = Circuit(2)
# Add some gates
c.add(gates.H(0))
c.add(gates.H(1))
# Define an initial state (optional - default initial state is |00>)
initial_state = np.ones(4) / 2.0
# Execute the circuit and obtain the final state
result = c(initial_state) # c.execute(initial_state) also works
print(result.state())
```



- QFT
- Grover
- QAOA
- VQE
- TSP
- QML models

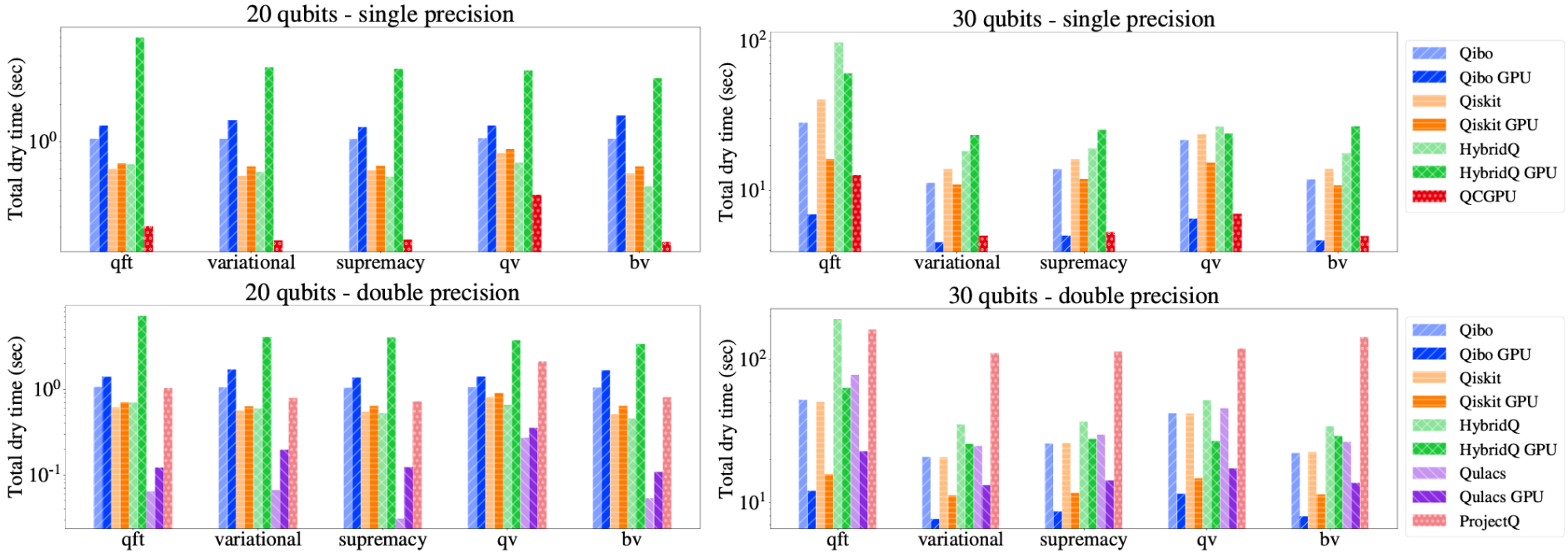
- Single qubit gates
- Two qubit gates
- Measurement gates
- Parametrized gates
- Multiple qubit gates
- Noise channels

- Numpy
- Tensorflow
- Pytorch
- Clifford
- CUDA
- Tensor Network

- Shots
- Probabilities
- Density matrices
- Gradients

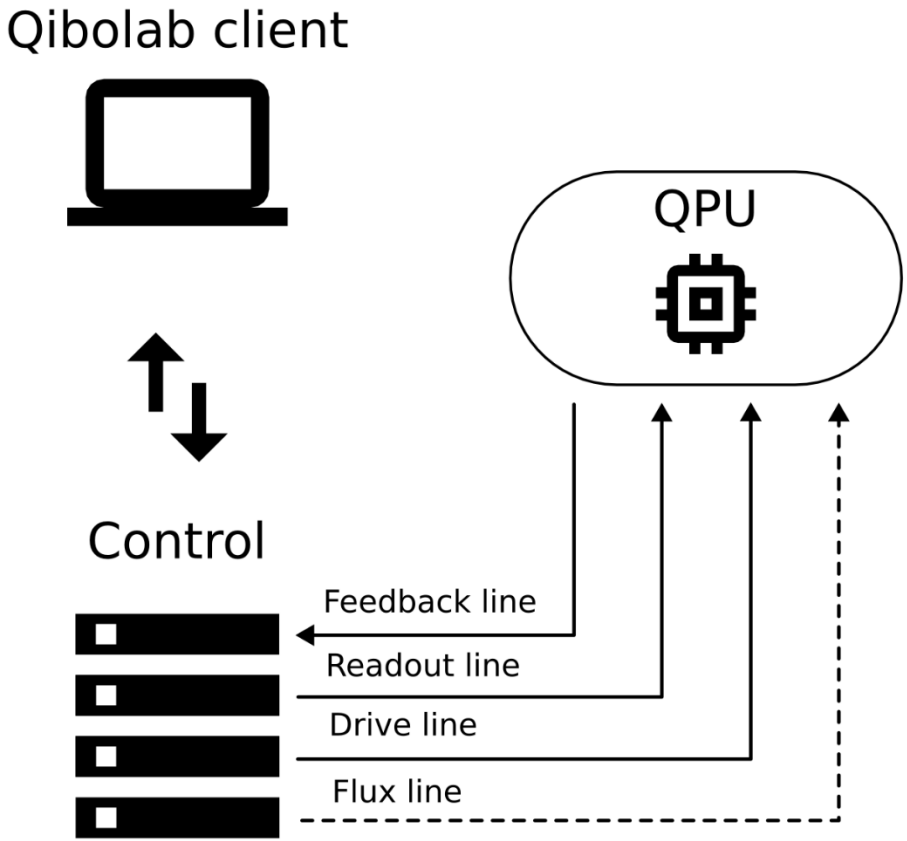
Qibo
QiboJit

Just In Time (JIT) compilation for optimal performances on both CPU and GPU.
Speedup for circuits with large number of qubits (>25).



QiboLab
Hardware control

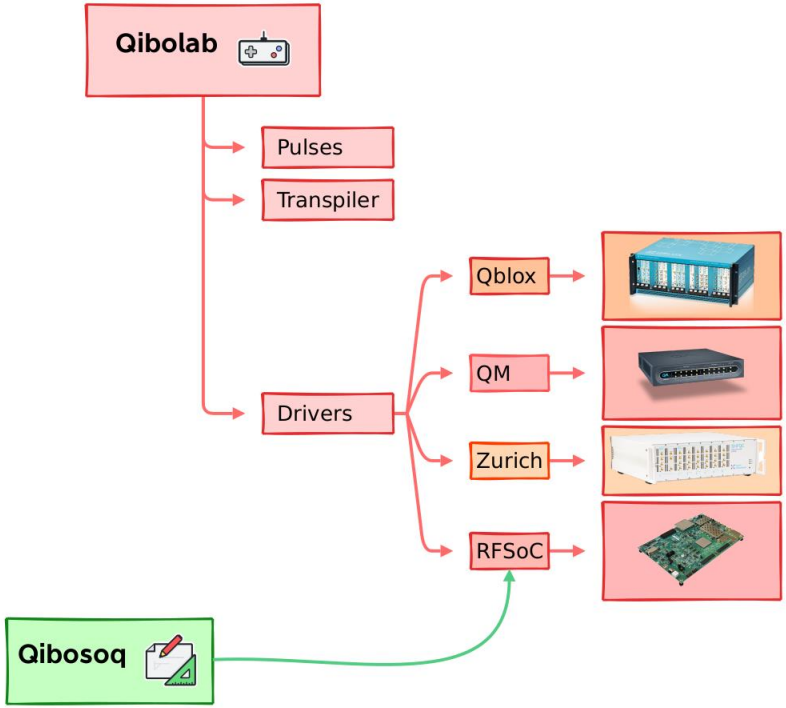
- The host computer running Qibolab communicates with the different electronics used to control a QPU.
- The readout and feedback channels measure the qubits.
- The drive channel applies gates.
- The flux channels allows for tuning qubit frequency to implement two qubit gates.



QiboLab
Software abstraction

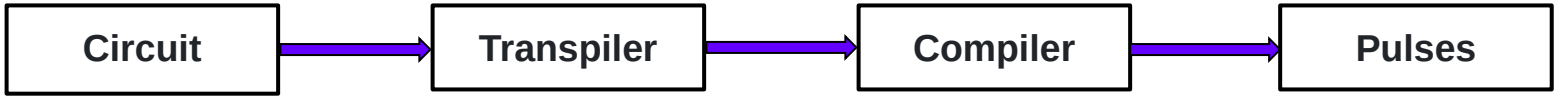
QiboLab provides two main interface objects:

- The Pulse for defining arbitrary pulses to be played on qubits.
- The Platform which is used to execute these pulses on a specific QPU and set of instruments.




```
"native_gates": {
    "single_qubit": {
        "0": {
            "RX": {
                "duration": 40,
                "amplitude": 0.456,
                "shape": "Gaussian(5)",
                "frequency": 4773712100,
                "relative_start": 0,
                "phase": 0,
                "type": "qd"
            },
```

QiboLab
From circuit to pulses

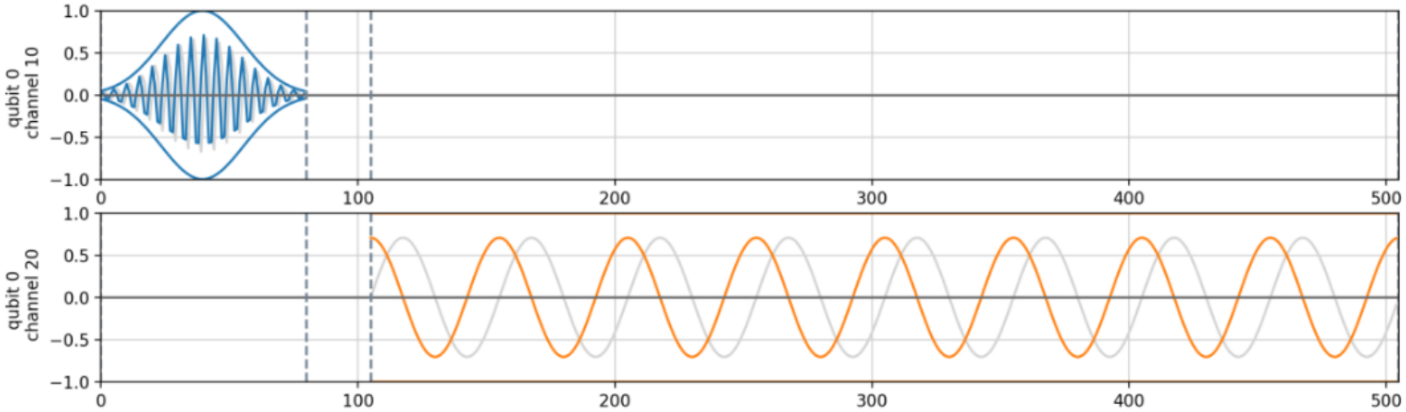


- Connectivity matching
- Initial qubit placing
- Native gates decomposition

- Frequency
- Phase
- Line

- Generation with qubit control instruments

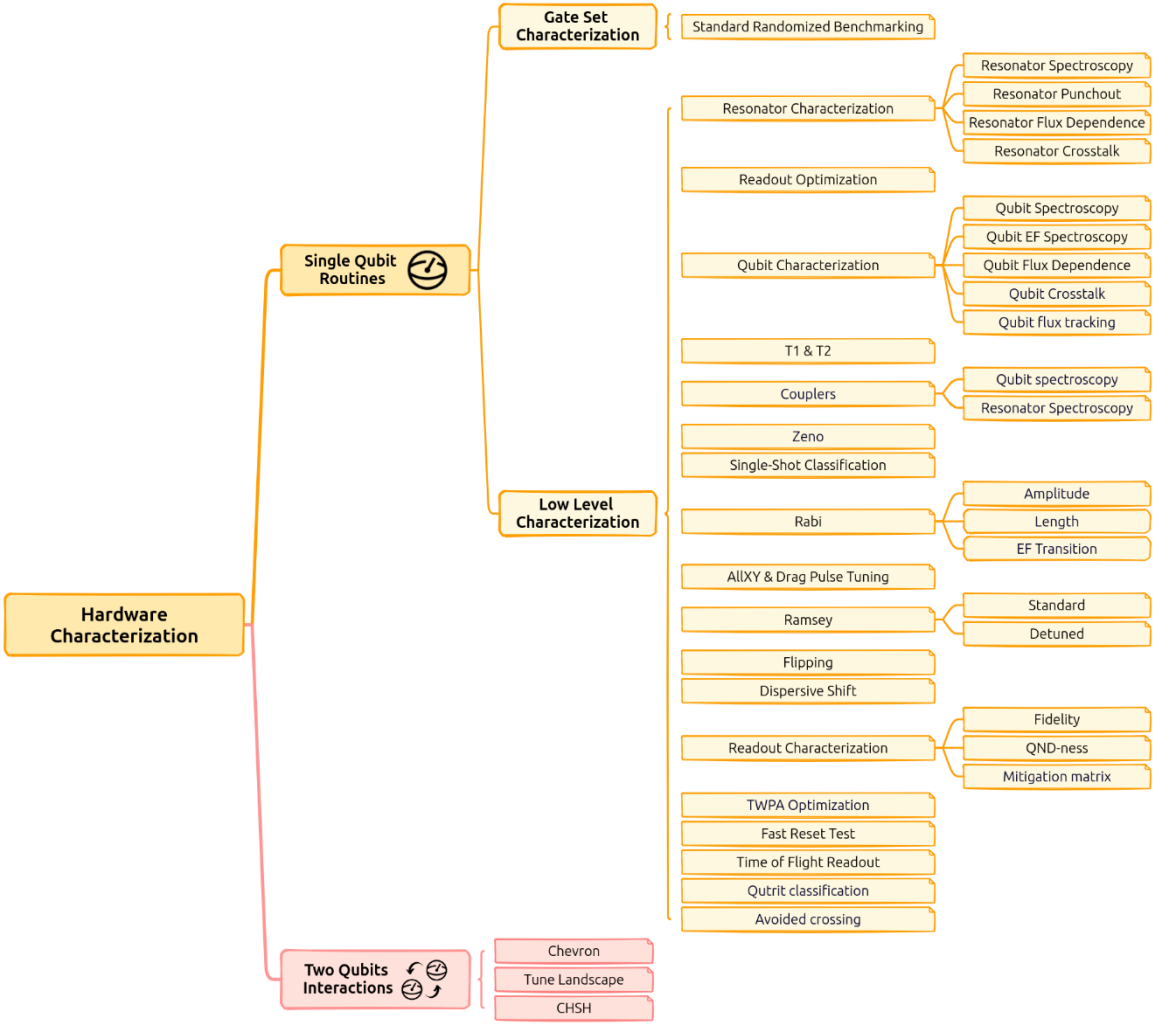
Conversion into pulses of a single qubit circuit composed of and RX gate followed by a measurement gate.



QiboCal
Qubit characterization

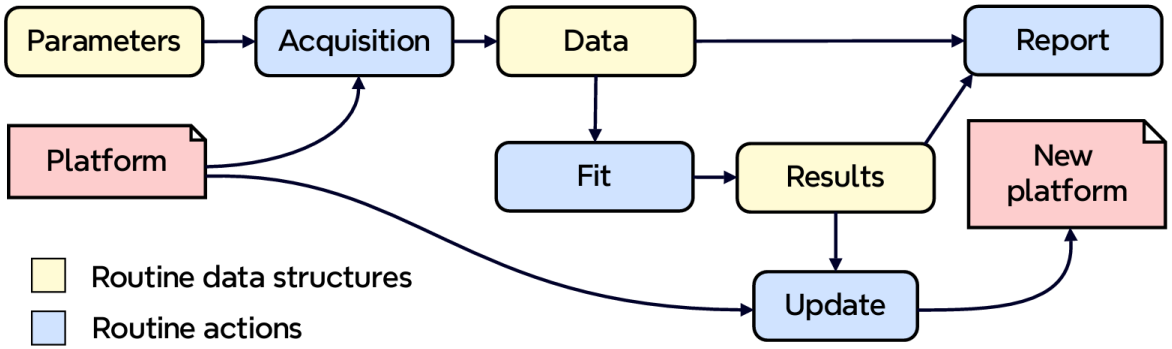
Find parameters of the platform and pulse objects used in QiboLab.
Main elements to be characterized:

- Readout resonator
- Single qubit gates
- Two qubit gates
- Performance benchmarking

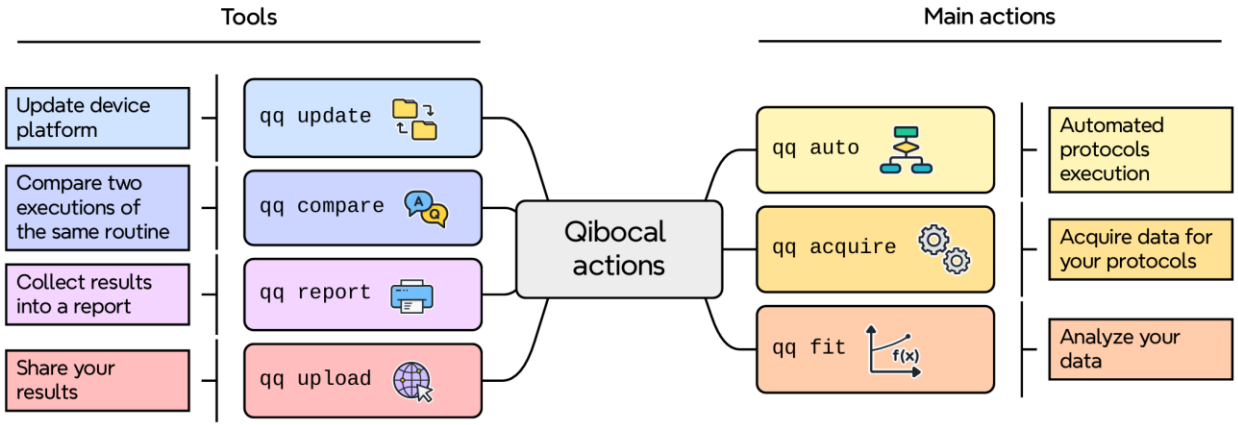


QiboCal
Calibration experiment

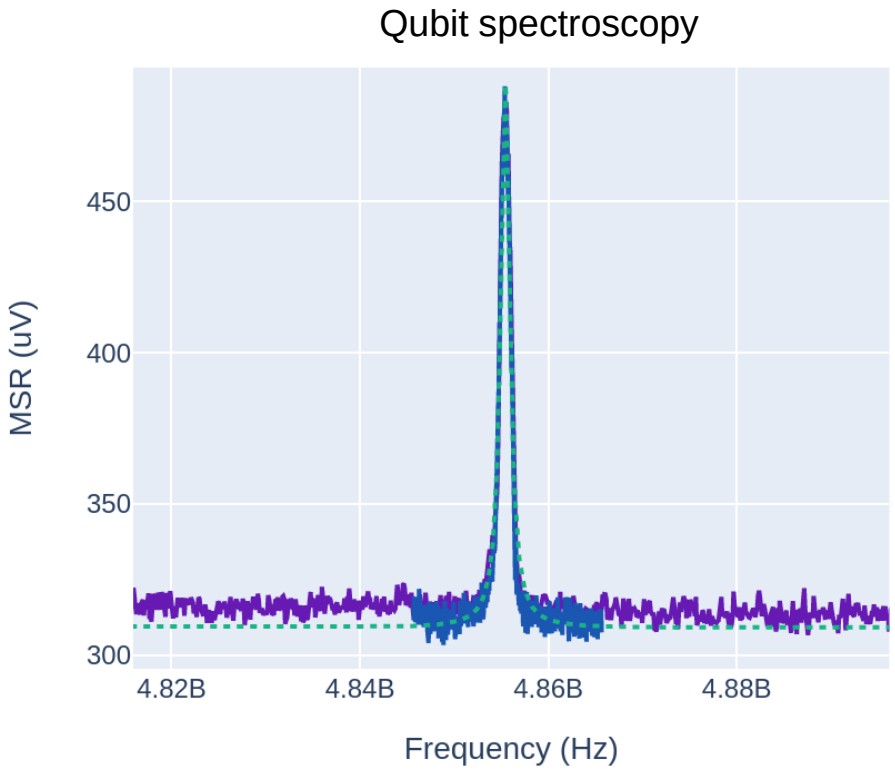
After the experiment a new platform is created by acquiring and fitting new data.

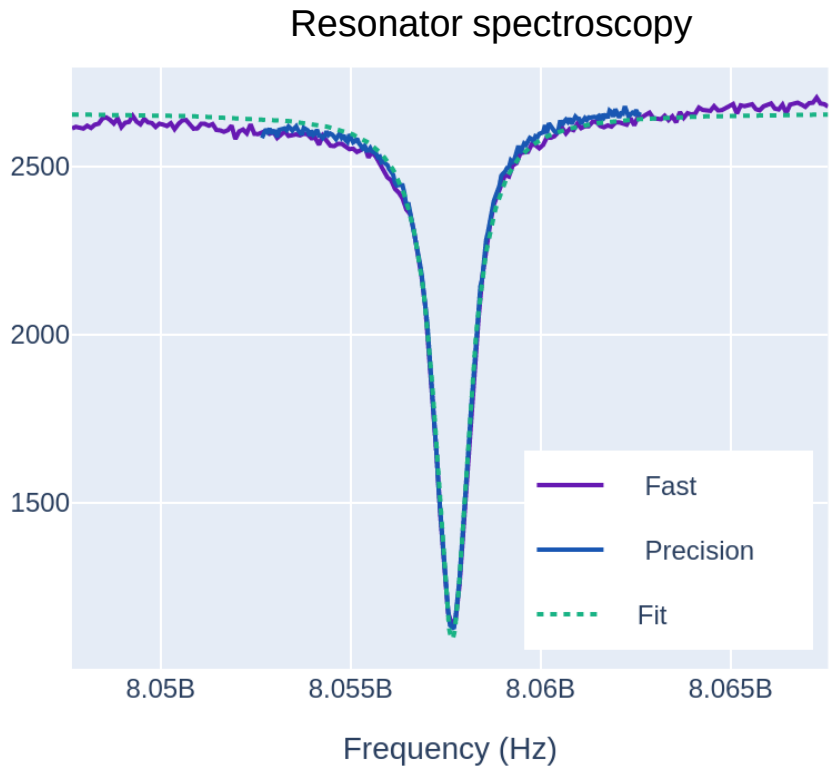


Main QiboCal commands to run an experiment.

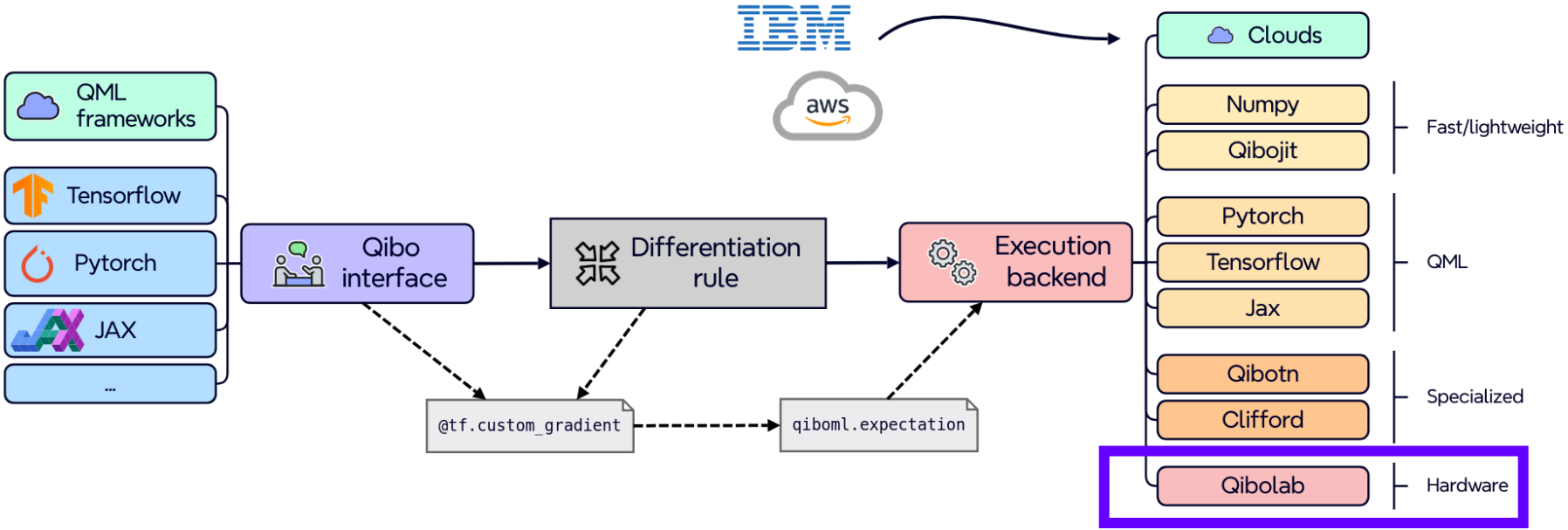


QiboCal
Calibration experiment





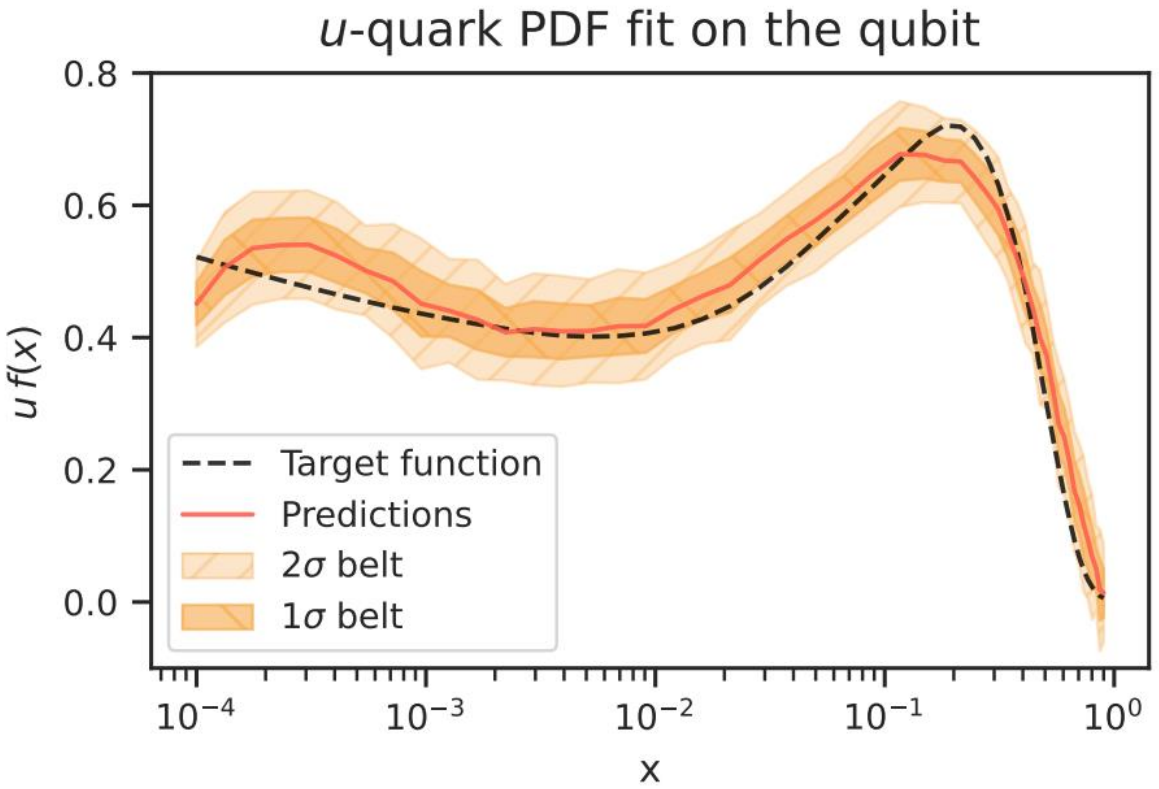
Under development
QiboML



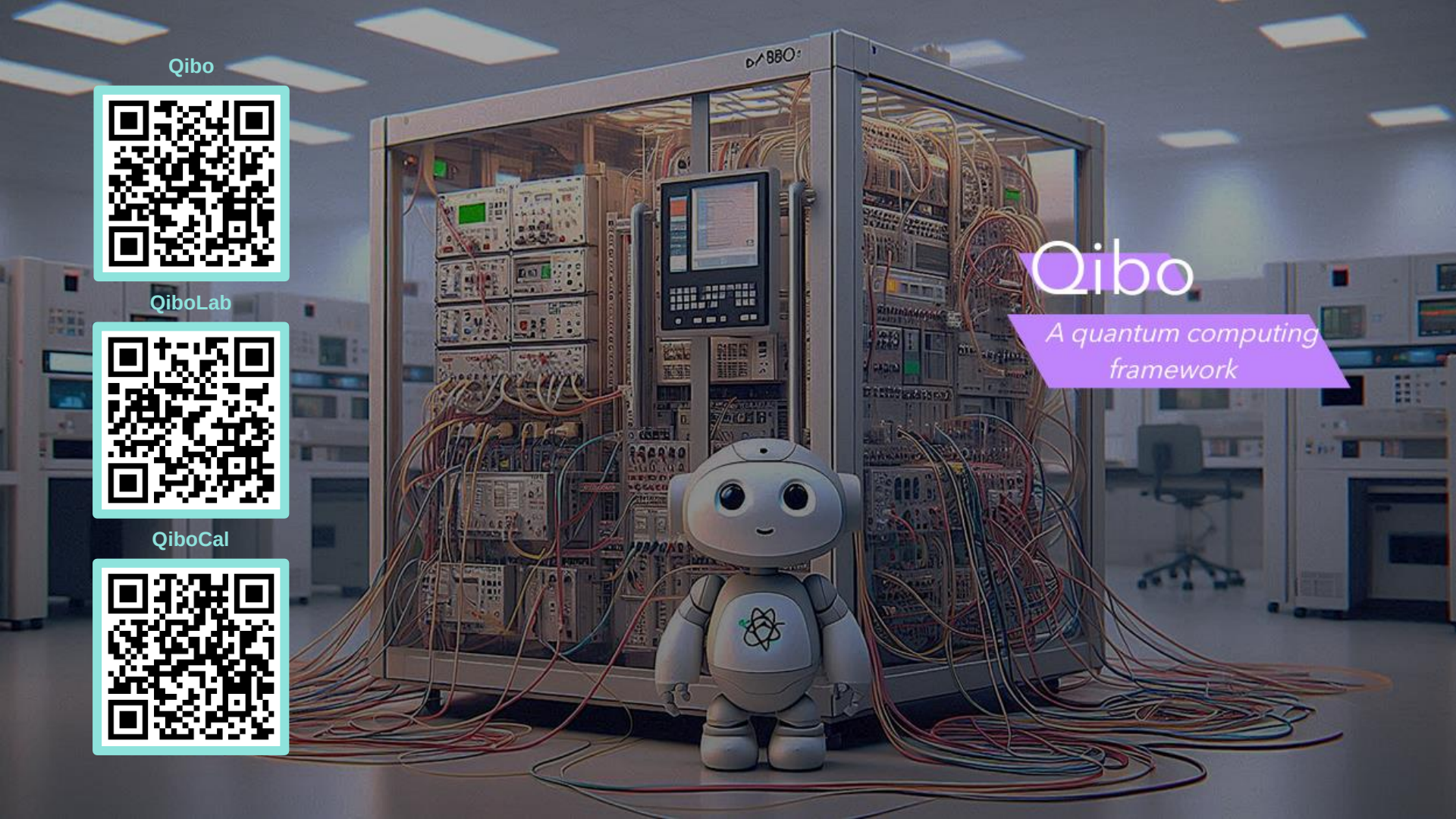
Toward full stack quantum machine learning

Under development
Full stack QML

Developing QML algorithms is particularly challenging in the NISQ era.
Qibo is the perfect environment to full-stack QML algorithms, from the high-level coding of the algorithm to the low-level deployment on the real qubits.



Estimates of 50 points of the u-quark PDF using 1-qubit device controlled by the RFSoC. The target values (black line) are compared with the estimates obtained with the qubit. The solid orange line and the confidence intervals are calculated by repeating 50 times the estimations with the trained model and then calculating means and standard deviation.


Qibo
QiboLab
QiboCal
Qibo
A quantum computing framework