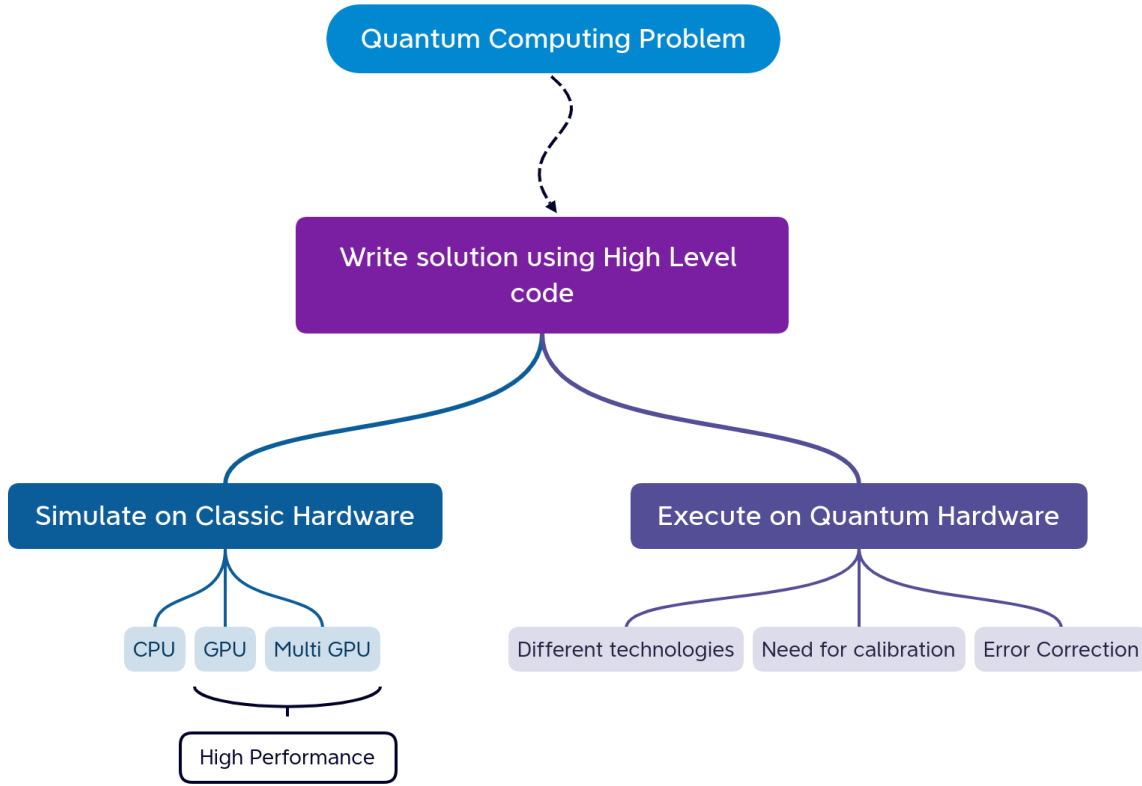
Towards a hybrid quantum operating system

Andrea Pasquale on the behalf of the Qibo collaboration

9th May 2023, CHEP2023, Norfolk


Los Alamos
NATIONAL LABORATORY
EST. 1943
CERN
QILIMANJARO
QUANTUM TECH
IFAE
Institut de Física d'Altes Energies
UNIVERSITAT DE BARCELONA
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
UNIVERSITÀ DEGLI STUDI DI MILANO
INFN
Istituto Nazionale di Fisica Nucleare
TII
Technology Innovation Institute
CQT
Centre for Quantum Technologies
National University of Singapore
SG NSCC SINGAPORE
National Supercomputing Centre
Agency for Science, Technology and Research
SINGAPORE
Universidade Federal do Rio de Janeiro
UBA
Universidad de Buenos Aires

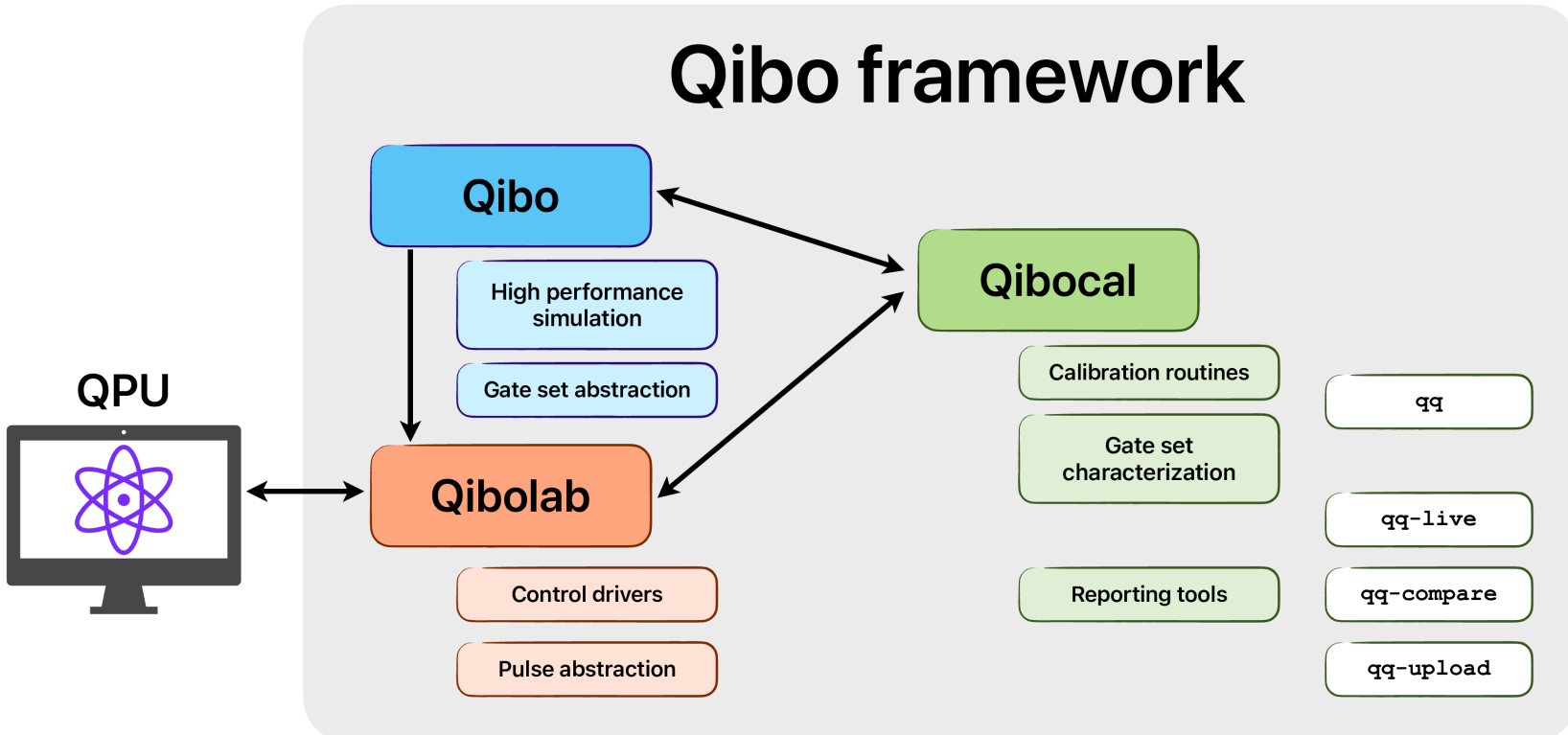
What is the challenge?



Is it possible to create from scratch a framework for all of this?

Introducing Qibo

Open-source full stack API for quantum simulation, hardware control and calibration


Qibo framework
Qibo
High performance simulation
Gate set abstraction
Qibocal
Calibration routines
Gate set characterization
Reporting tools
qq
qq-live
qq-compare
qq-upload
QPU
Qibolab
Control drivers
Pulse abstraction

High performance simulation

Gate set abstraction

```python
import numpy as np
from qibo.models import Circuit
from qibo import gates, set_backend

# Set driver engine
set_backend("numpy")

c = Circuit(2)
c.add(gates.X(0))

# Add a measurement register on both qubits
c.add(gates.M(0, 1))

# Execute the circuit with the default initial state |00>.
result = c(nshots=100)
```

```python
# Change backend
set_backend("qibojit")

# Circuit execution with new driver
result = c(nshots=100)
```

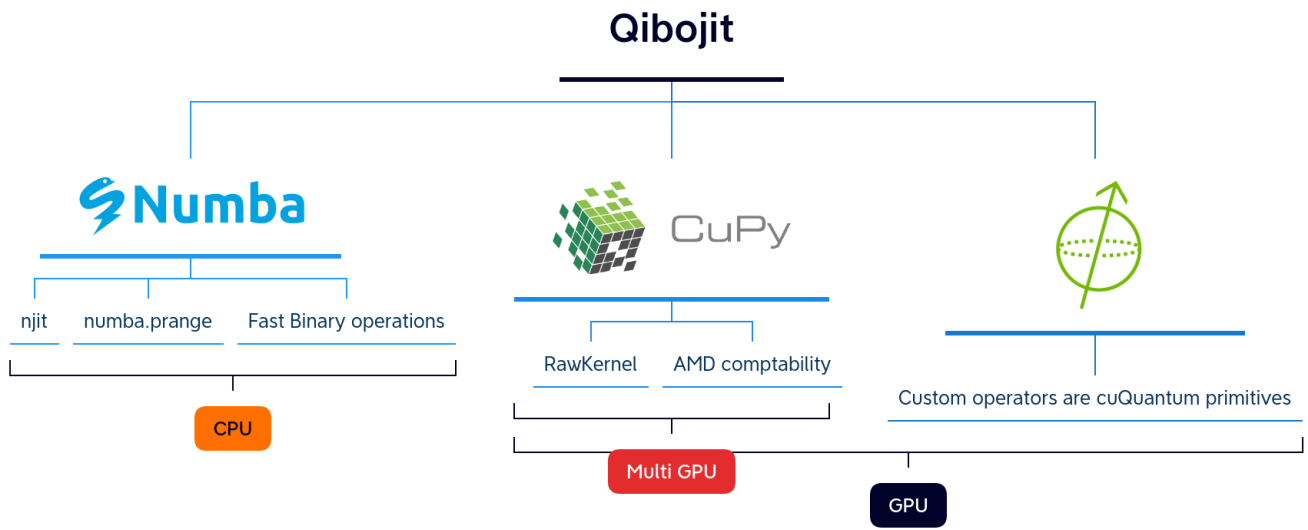
Qibo features

- Definition of a standard language for the construction and execution of quantum circuits with device agnostic approach to simulation and quantum hardware control based on plug and play backend drivers.
- A continuously growing code-base of quantum algorithms applications presented with examples and tutorials.
- Efficient simulation backends with GPU, multi-GPU and CPU with multi-threading support.
- A Simple mechanism for the implementation of new simulation and hardware backend drivers.

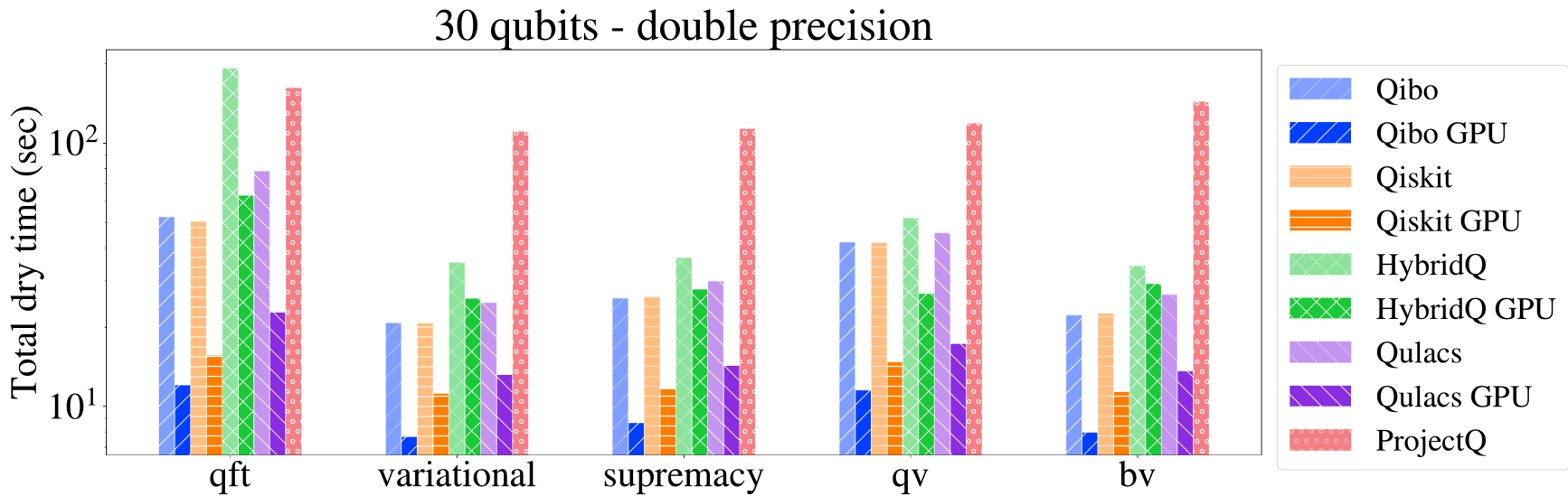
High performance simulation

❌ Long computational times using naive approach (`Numpy` or `TensorFlow`) for circuits with large number of qubits.

✅ We need more sophisticated tools to be able to simulate a quantum circuits with more qubits!



BENCHMARK



All the benchmarks are available in qibojit-benchmarks.

Hardware Control using Qibo

How are gates implemented on a Quantum Computer?

↓

By sending microwave pulses.

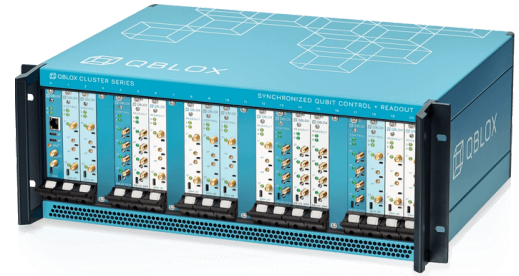
↓

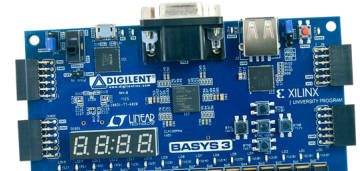
How do we control them?

↓

Using FPGAs

→ We need a Pulse API + Drivers for different electronics.





Introducing Qibolab

Automatic deployment of quantum circuits on quantum hardware

What is Qibolab?

Introducing Qibocal

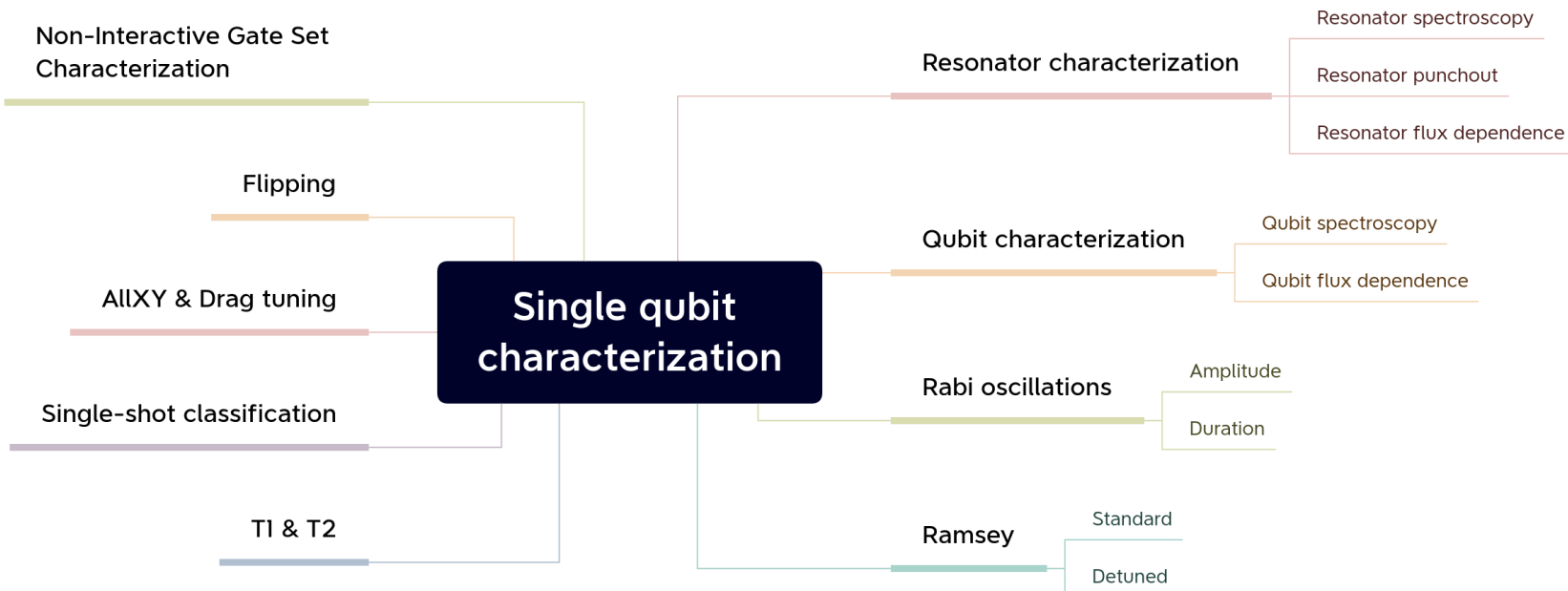
A reporting tool for calibration using Qibo

Motivation

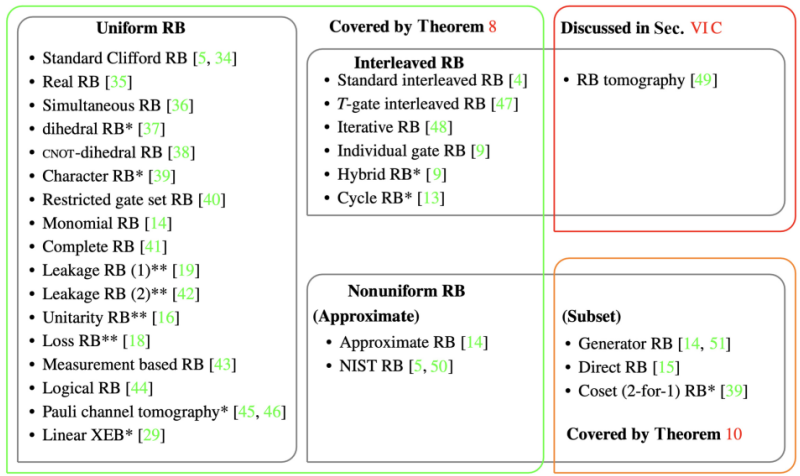
Suppose that we have assembled a quantum computer and we have a way to send pulses to the chip... are we done? NO

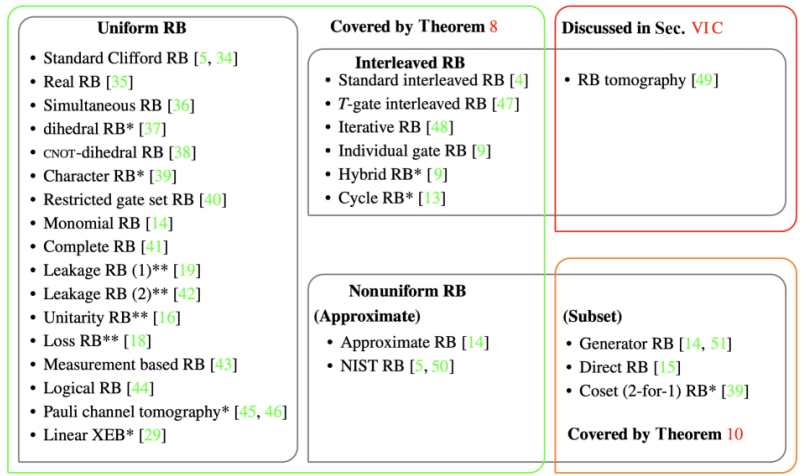
We need to characterize, validate and verificate our qubits.

Single Qubit Characterization: Pulse Level



Single Qubit Characterization: Circuit Level

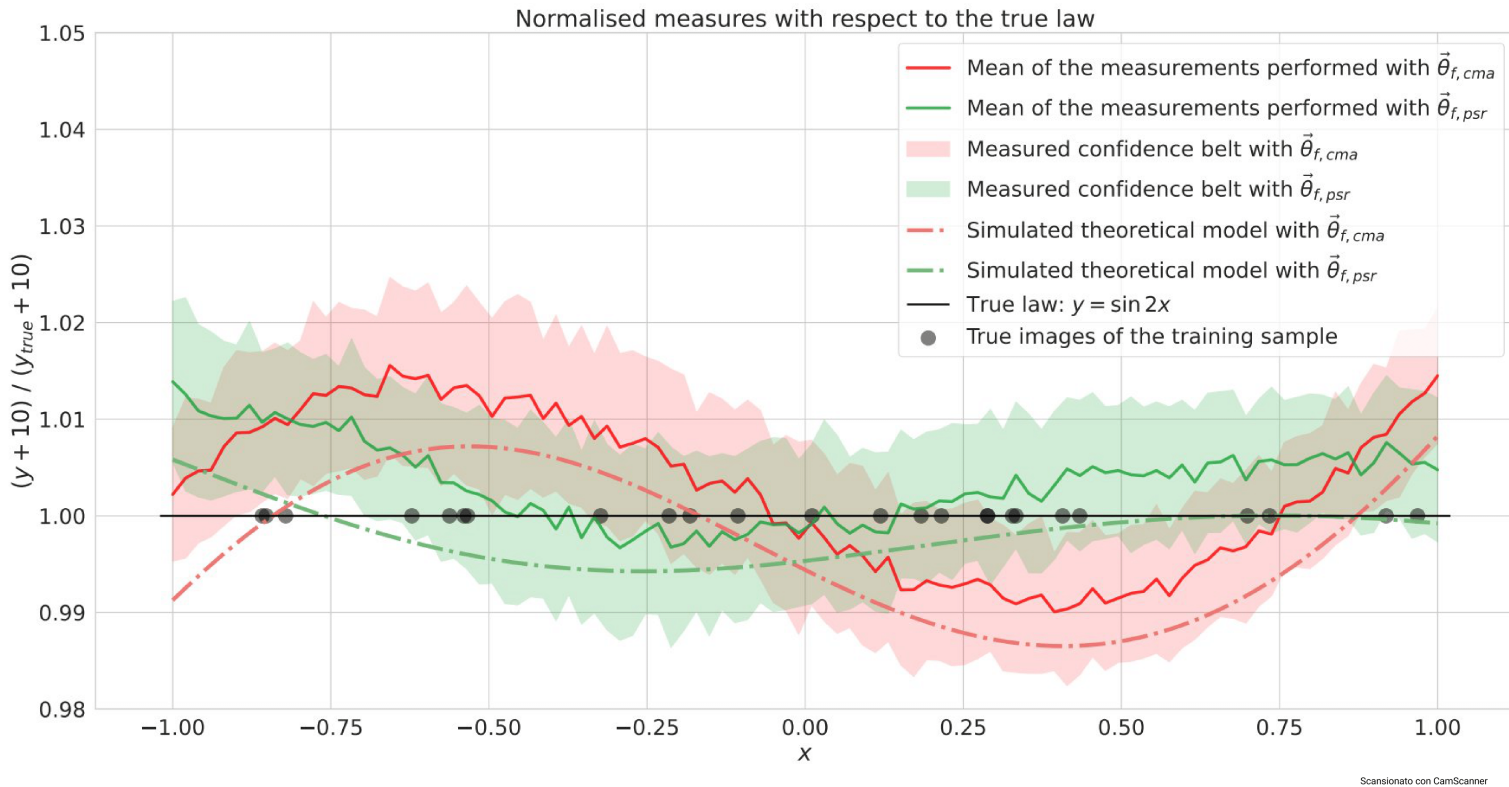




Applications

What can we achieve using `Qibo` + `Qibolab` + `Qibocal`?

Successfully performed a gradient descent on a QPU with a single using Parameter Shift Rule algorithm.



Outlook

Qibo is growing to accomodate different tasks:

- High performance quantum simulation: qibojit
- Hardware control: qibolab
- Hardware calibration: qibocal

What makes Qibo different from the other libraries:

- Publicly available as an open source project
- Modular layout design with the possibility of adding
    - new backends for simulation
    - new backends for hardware control
- Community driven effort

Thanks for listening!