

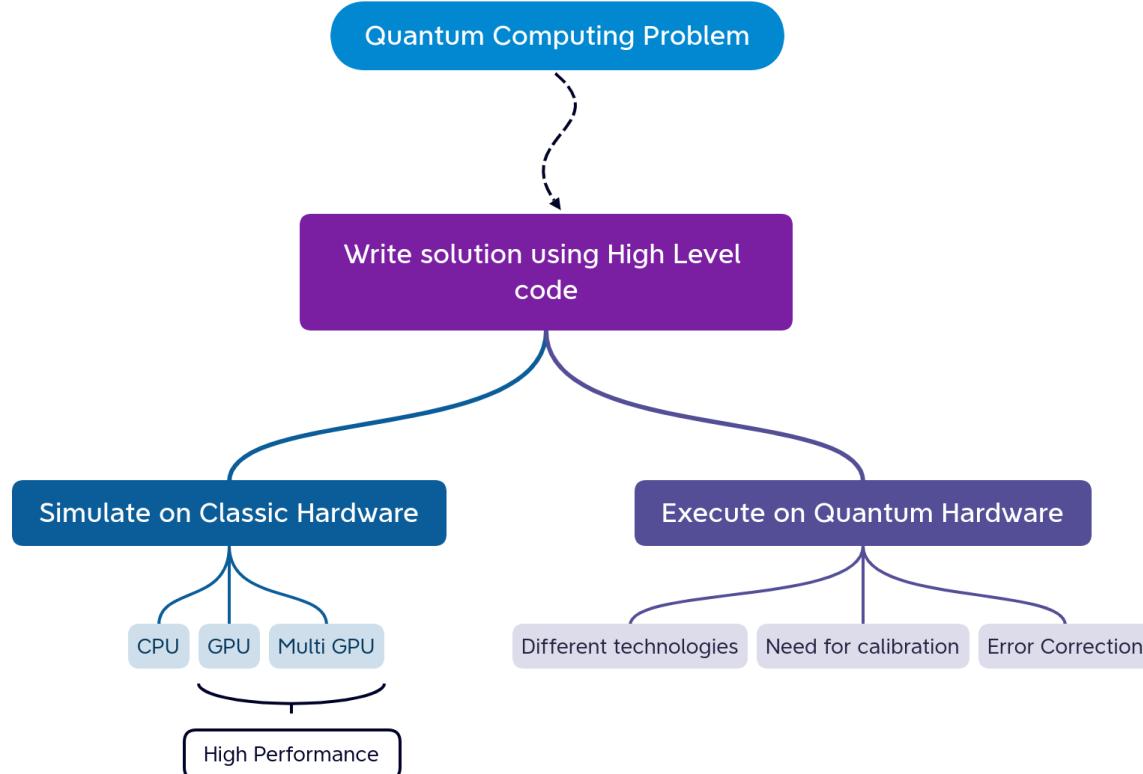
TOWARDS A HYBRID QUANTUM OPERATING SYSTEM

Andrea Pasquale on the behalf of the Qibo collaboration

9th May 2023, CHEP2023, Norfolk



WHAT IS THE CHALLENGE?

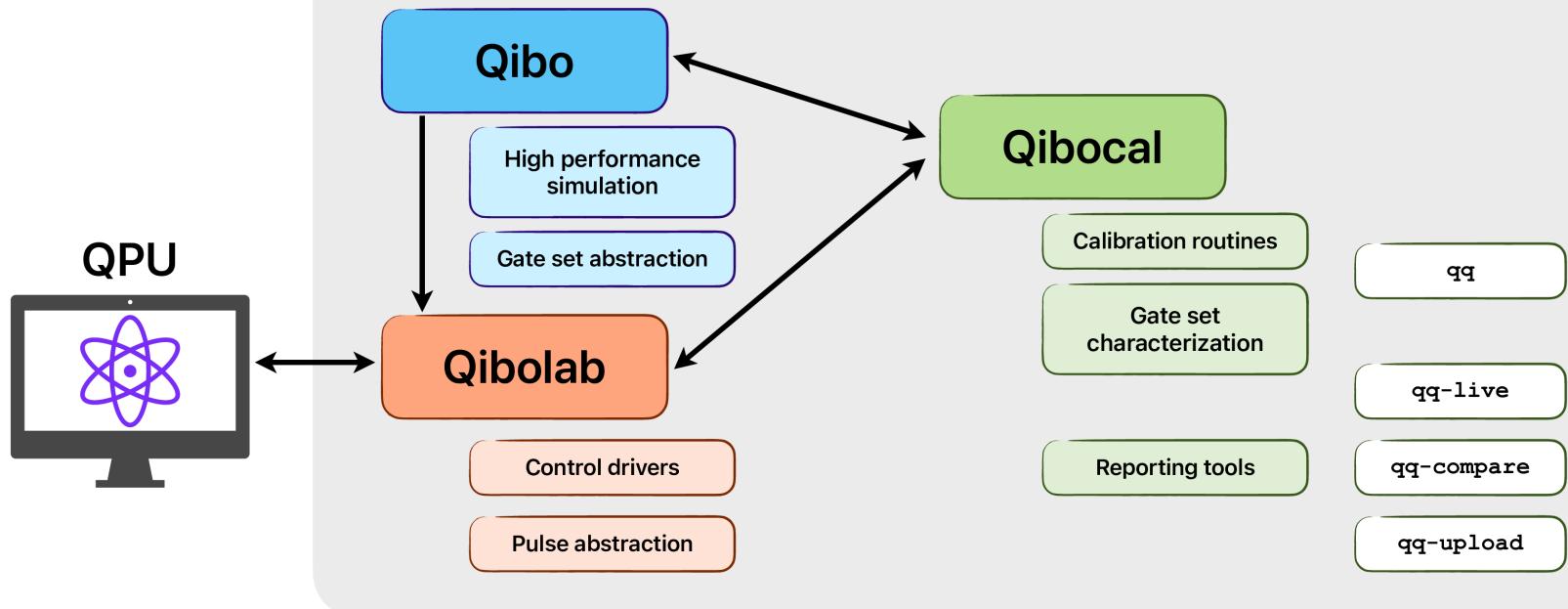


Is it possible to create from scratch a framework for all of this?

INTRODUCING QIBO

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

Qibo framework



SIMULATION

GATE SET ABSTRACTION

```
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)

# Change backend
set_backend("qibojit")

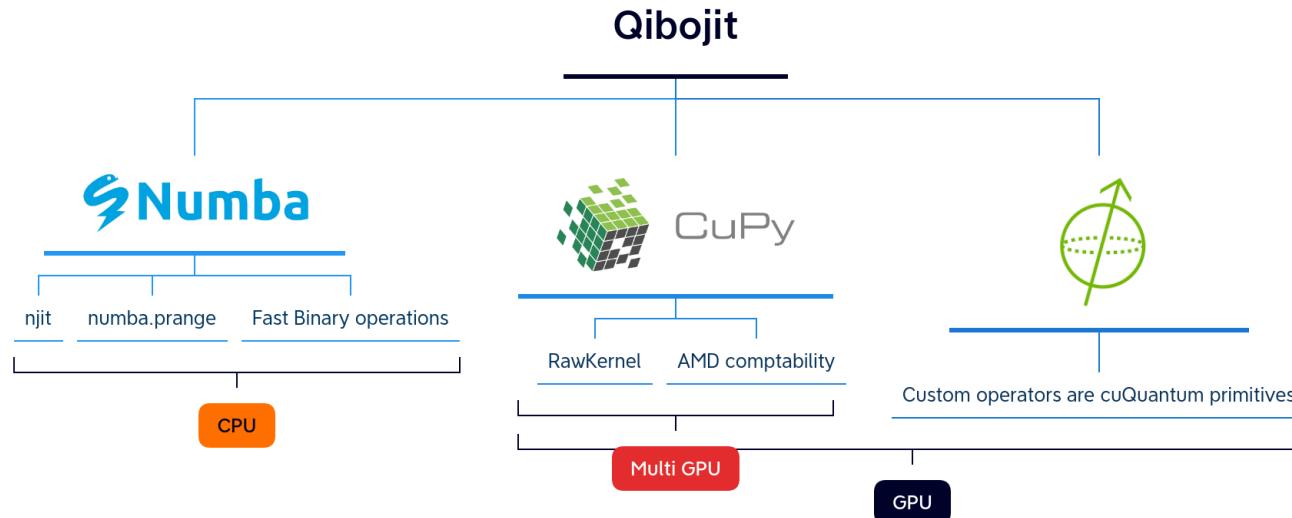
# Circuit execution with new driver
result = c(nshots=100)
```

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**.

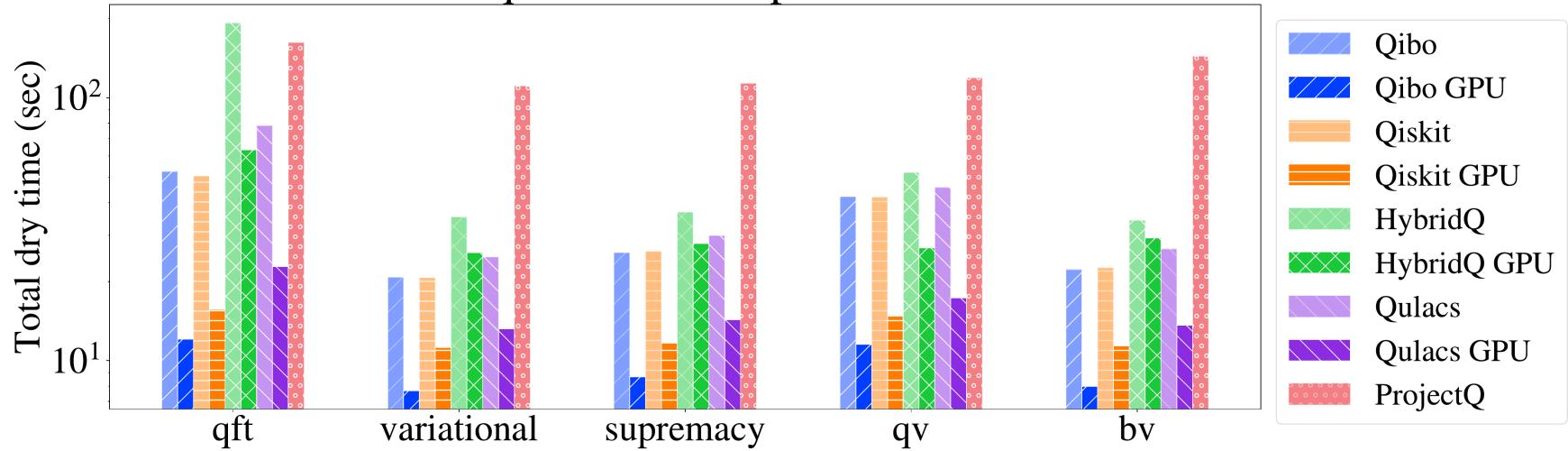
HIGH PERFORMANCE SIMULATION

- ✗ 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

30 qubits - double precision



All the benchmarks are available in [qibojit-benchmarks](#).

QIBOLAB

Automatic deployment of quantum circuits on quantum hardware

MOTIVATION

How are gates implemented on a quantum computer?

By sending microwave pulses.

How do we control them?

Using FPGAs

→ We need both a pulse level API + drivers to interface Qibo with different instruments.



PULSE API EXAMPLE

```
from qibolab import Platform
from qibolab.pulses import ReadoutPulse, PulseSequence

# Define PulseSequence
sequence = PulseSequence()
# Add some pulses to the pulse sequence
sequence.add(ReadoutPulse(start=0,
                           amplitude=0.3,
                           duration=4000,
                           frequency=200_000_000,
                           shape='Gaussian(5)'))

# Define platform
platform = Platform("tii1q_b1")
# Platform setup
platform.connect()
platform.setup()
platform.start()
# Executes a pulse sequence.
results = platform.execute_pulse_sequence()
platform.stop()
platform.disconnect()
```

DRIVERS IMPLEMENTED

Currently Qibolab supports the following drivers:

- Qblox
- Quantum Machines
- Zurich Instruments
- RFSoC
- ZCU11 (under development)

We also supports local oscillators

- RohdeSchwarz SGS100A
- ERASynth

INTRODUCING QIBOCAL

A reporting tool for calibration using Qibo

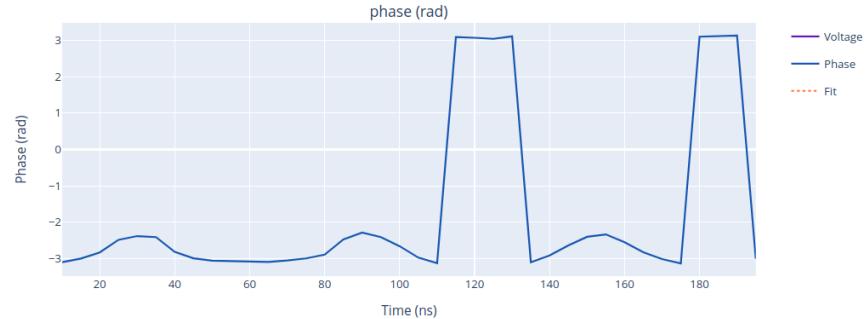
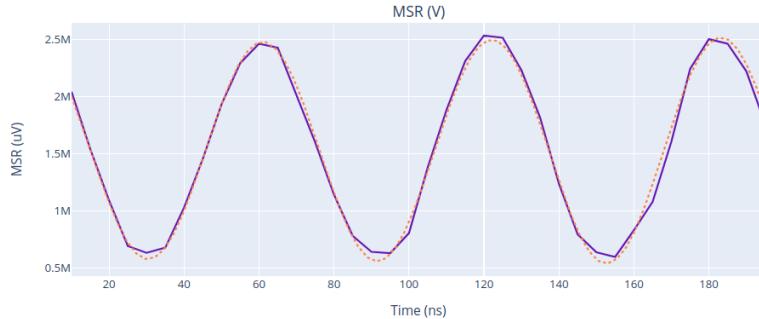
MOTIVATION

Let's suppose the following:

1. We have a QPU (self-hosted).
2. We have control over what we send to the QPU.
3. We know how to convert Pulses to Circuits.

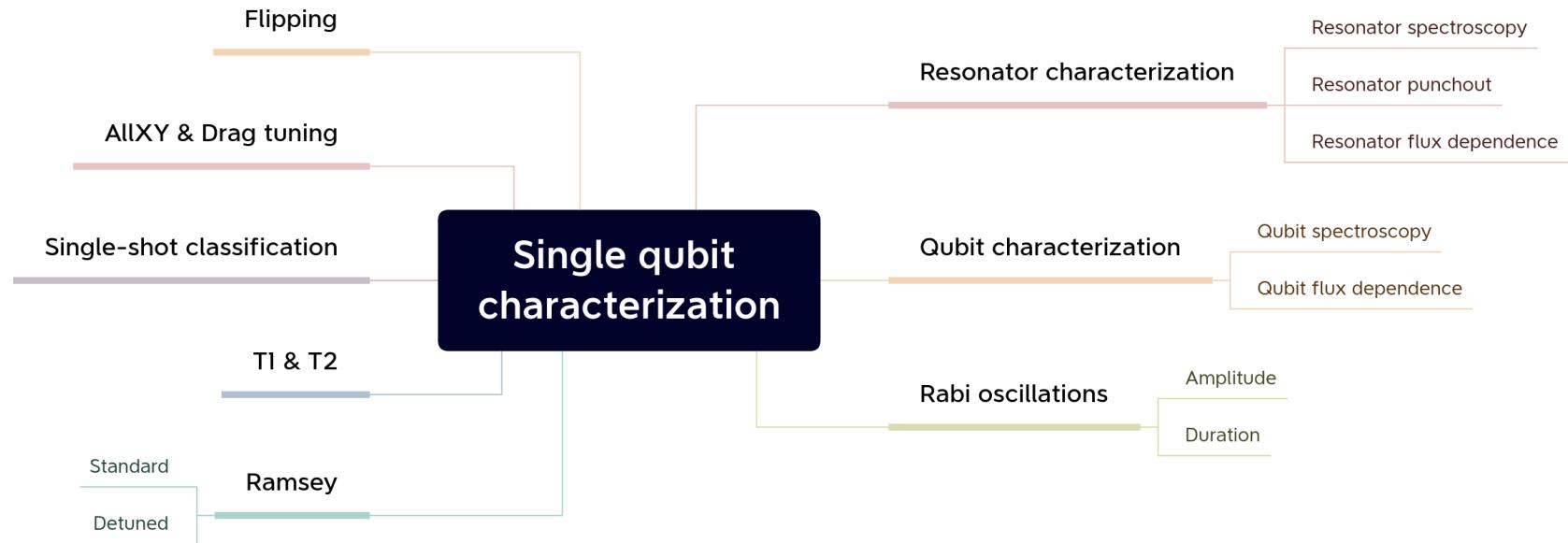
Can I **trust** my results? NO!

Characterization and **calibration** are an essential step to properly operate emerging quantum devices.



Calibration of RX pulse amplitude through a Rabi experiment through Qibocal.

SINGLE QUBIT CHARACTERIZATION: PULSE LEVEL



SINGLE QUBIT CHARACTERIZATION: CIRCUIT LEVEL

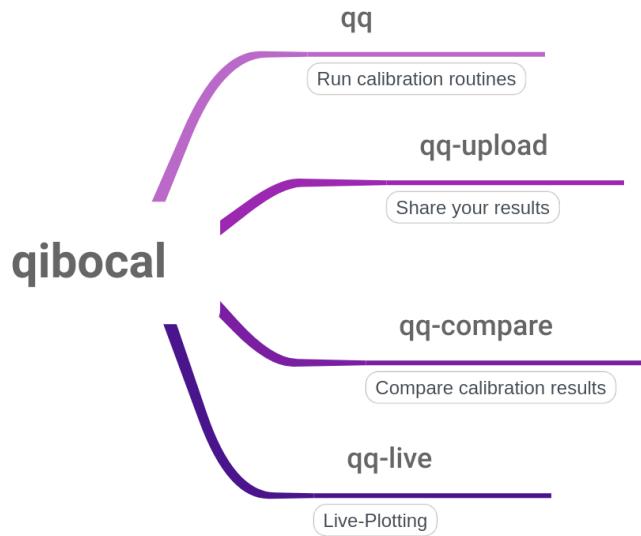
Uniform RB	Covered by Theorem 8	Discussed in Sec. VI C
<ul style="list-style-type: none">• Standard Clifford RB [5, 34]• Real RB [35]• Simultaneous RB [36]• dihedral RB* [37]• CNOT-dihedral RB [38]• Character RB* [39]• Restricted gate set RB [40]• Monomial RB [14]• Complete RB [41]• Leakage RB (1)** [19]• Leakage RB (2)** [42]• Unitarity RB** [16]• Loss RB** [18]• Measurement based RB [43]• Logical RB [44]• Pauli channel tomography* [45, 46]• Linear XEB* [29]	<ul style="list-style-type: none">• Interleaved RB• Standard interleaved RB [4]• T-gate interleaved RB [47]• Iterative RB [48]• Individual gate RB [9]• Hybrid RB* [9]• Cycle RB* [13]	<ul style="list-style-type: none">• RB tomography [49]
<ul style="list-style-type: none">• Nonuniform RB (Approximate)• Approximate RB [14]• NIST RB [5, 50]		<p>(Subset)</p> <ul style="list-style-type: none">• Generator RB [14, 51]• Direct RB [15]• Coset (2-for-1) RB* [39] <p>Covered by Theorem 10</p>

Currently in Qibocal the following protocols are implemented:

- Standard RB
- Simulataneous Filtered RB
- XId RB

We are currently developing a suite for the development of the **latest** quantum benchmarking protocols available in the **literature**.

REPORTING TOOLS



- Platform agnostic approach
- Launch calibration routines easily
- Live-plotting tools
- Autocalibration (under development)

Home

Timestamp

Summary

Actions

Resonator Spectroscopy Low Power - 0

Qubit Spectroscopy - 0

Rabi - 0

Ramsey - 0

Standardrb - 0

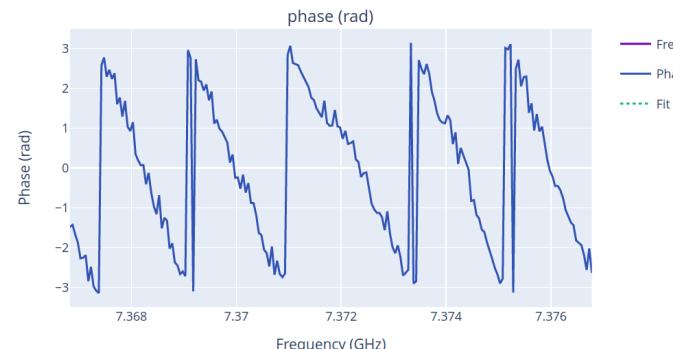
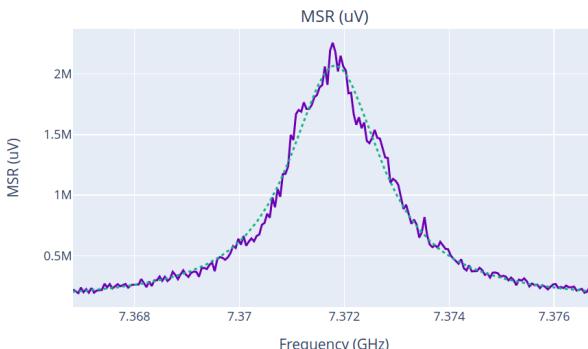
Actions

Please find below data generated by actions:

Resonator Spectroscopy Low Power - 0

- Qubit 0

qubit	Fitting Parameter	Value
0	readout frequency	7,371,823,304 Hz
0	amplitude	0.022

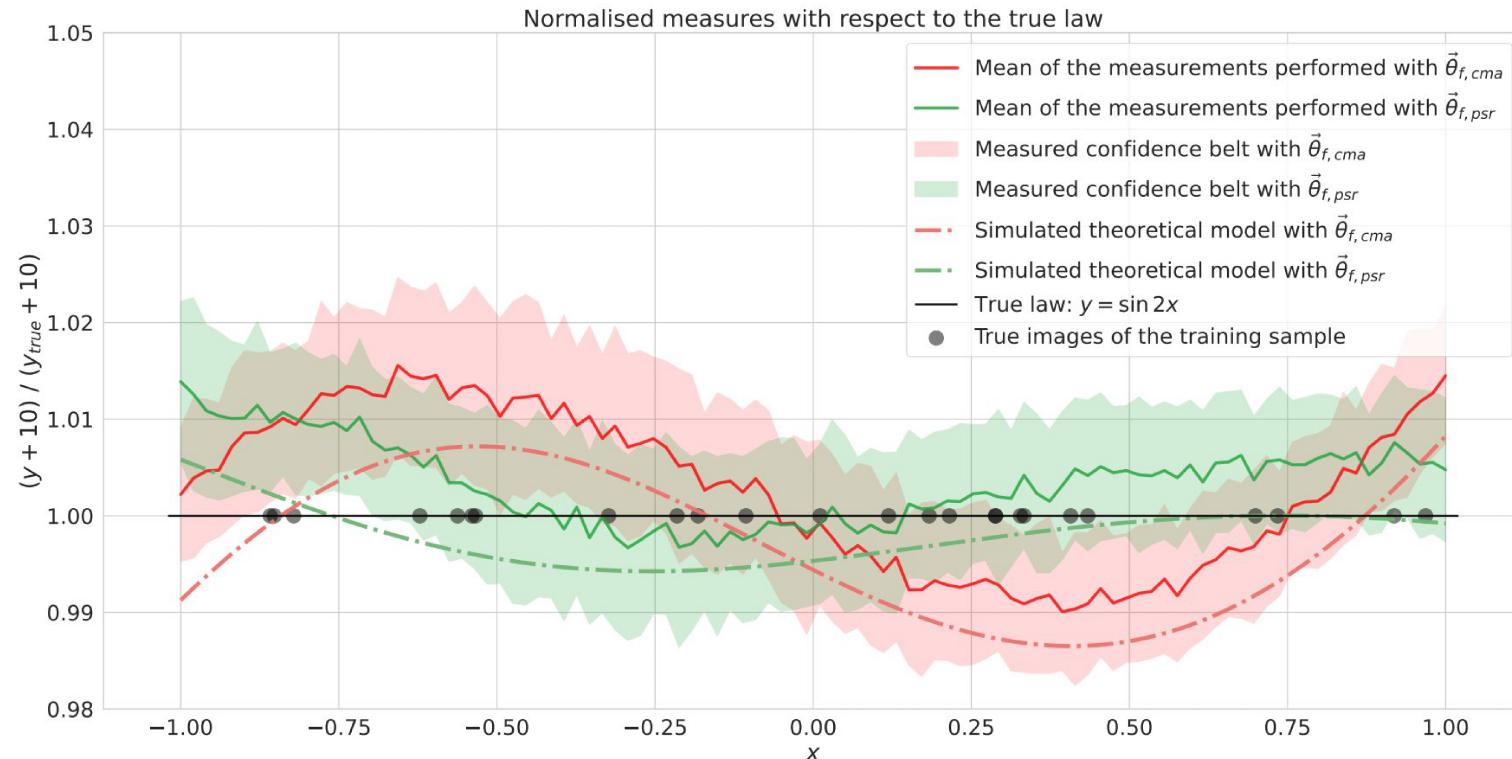


APPLICATIONS

PARAMATER SHIFT RULE ON HARDWARE

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

Why should you choose Qibo?

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

The screenshot shows the GitHub repository page for 'qibo' (Public). The repository has 14 branches and 26 tags. The 'About' section describes it as a framework for quantum computing with hardware acceleration, mentioning GPU, quantum, quantum-computing, quantum-circuit, quantum-algorithms, and quantum-annealing. It includes links for Readme, Apache-2.0 license, stars (185), watching (24), forks (42), and reporting issues. The 'Releases' section shows Qibo 0.1.13 (Latest) was released last week. The 'Packages' section lists 'qibo'. The 'Used by' section shows 6 projects using Qibo, with logos for Cirq, Qiskit, Q#, Quirk, and OpenQASM. The 'Contributors' section shows 29 contributors with their GitHub profiles. The 'Environments' section shows a 'github-pages' environment as active.

qibo Public

master 14 branches 26 tags Go to file Add file Code About

A framework for quantum computing with hardware acceleration.

qibo.science

gpu quantum quantum-computing quantum-circuit quantum-algorithms quantum-annealing

Readme Apache-2.0 license

185 stars 24 watching 42 forks Report repository

Releases 24

Qibo 0.1.13 (Latest) last week + 23 releases

Packages 1

qibo

Used by 6

Cirq Qiskit Q# Quirk OpenQASM

Contributors 29

+ 18 contributors

Environments 1

github-pages (Active)

scarrazza Merge pull request #877 from qiboteam/Add-reference-to-ano... 2582f46 2 days ago 6,224 commits

.github add poetry-extras input 2 months ago

doc fix examples 3 days ago

docker adding docker deployment for tags 2 years ago

examples Add reference to anomaly detection example 2 days ago

src/qibo No commit message 3 days ago

tests [pre-commit.c] auto fixes from pre-commit.com hooks 3 days ago

.gitignore [pre-commit.c] auto fixes from pre-commit.com hooks 6 months ago

.pre-commit-config.yaml [pre-commit.c] pre-commit autoupdate last week

.readthedocs.yml trying poetry with rtd 2 months ago

LICENSE Update LICENSE 3 years ago

README.md updating links 5 months ago

.poetry.lock delete otional input last month

pyproject.toml updating version last week

README.md

QIBO

Tests passing codecov 100% Docs passing DOI 10.5281/zenodo.7868238

Qibo is an open-source full stack API for quantum simulation and quantum hardware control.

Some of the key features of Qibo are:

- 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.
- Simple mechanism for the implementation of new simulation and hardware backend drivers.

THANKS FOR LISTENING!