TOWARDS A HYBRID QUANTUM OPERATING SYSTEM

Andrea Pasquale on the behalf of the Qibo collaboration

9th May 2023, CHEP2023, Norfolk

Why Quantum Computing in HEP?

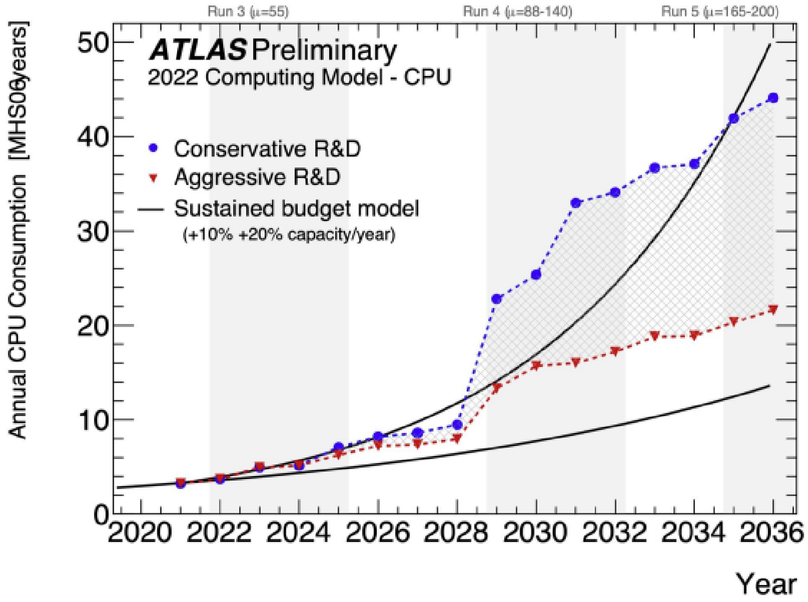
Nat Rev Phys 4, 143–144

LHC produces TB of data per second which need to processed.

Currently we cannot keep up with the computational resources required!

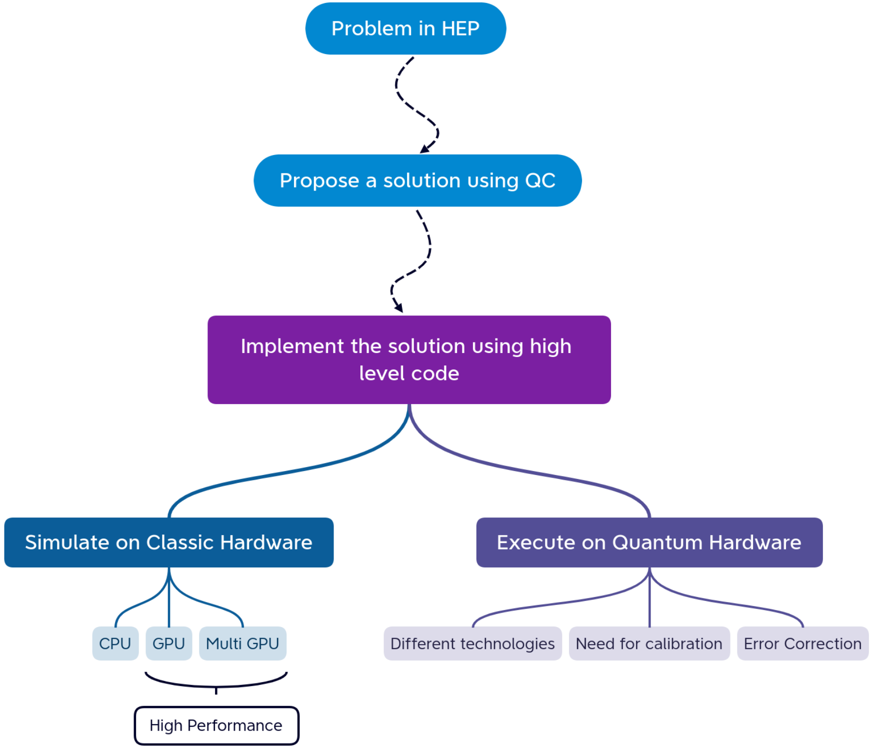
Quantum Computing (QC), especially Quantum Machine Learning (QML), is a promising solution:

- Efficient in high-dimensional quantum state spaces
- Potential computational speed-ups
- Qubit representation offers better compression
- Acceleration of classically expensive operations (eigensolvers).



Projected compute usage from ATLAS Software and Computing HC-LHC Roadmap

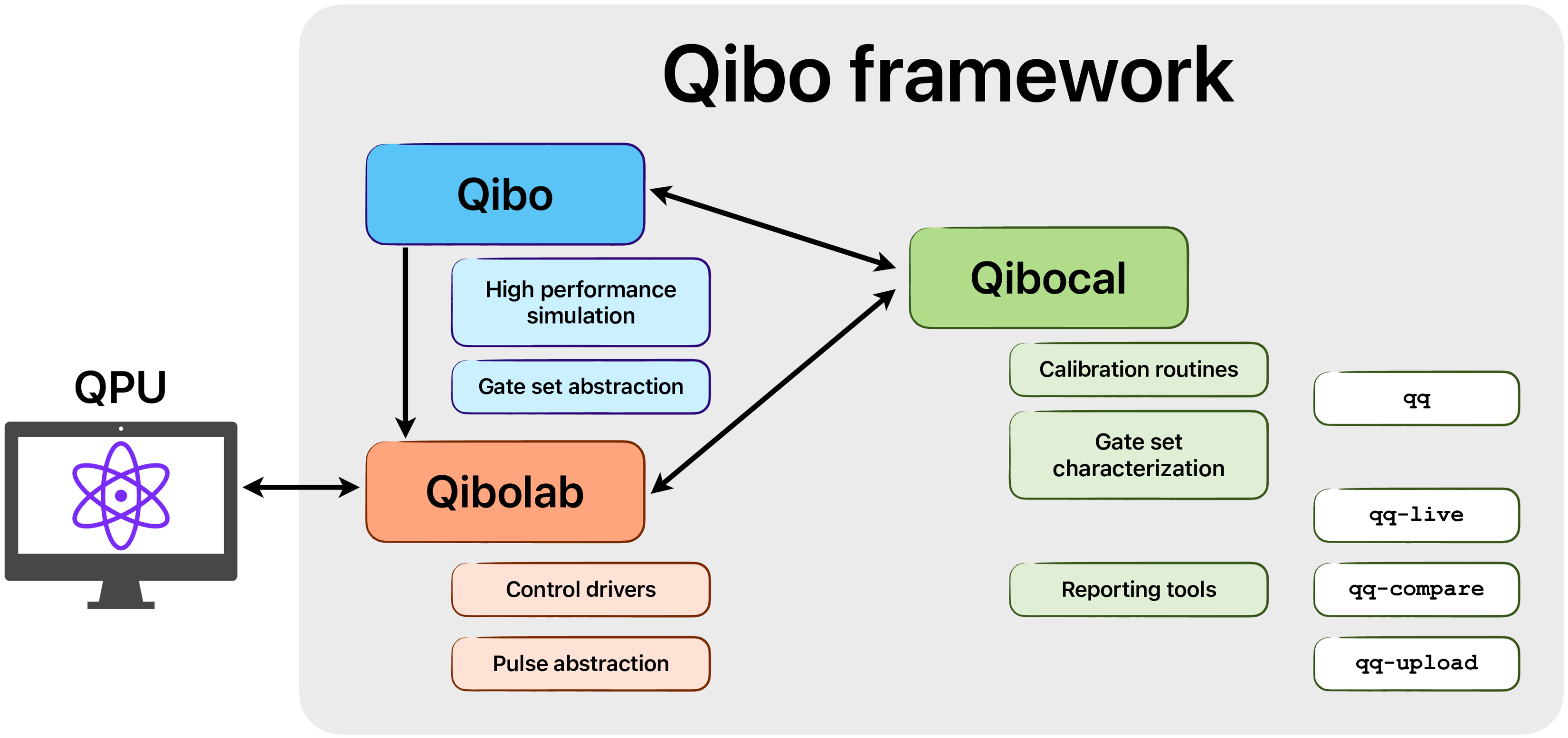
How can we start using Quantum Computing?



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

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

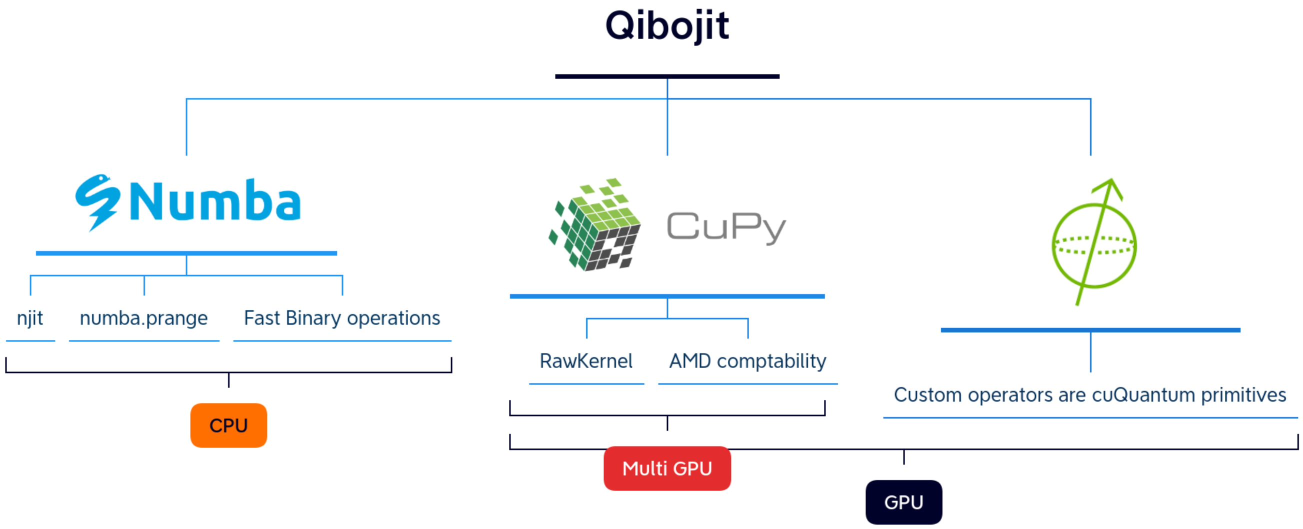
- Definition of a standard language for the construction and execution of quantum circuits with device agnostic approach to simulation and quantum hardware control based on plug and play backend drivers.
- A continuously growing code-base of quantum algorithms and applications presented with examples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

2009.01845

High performance simulation

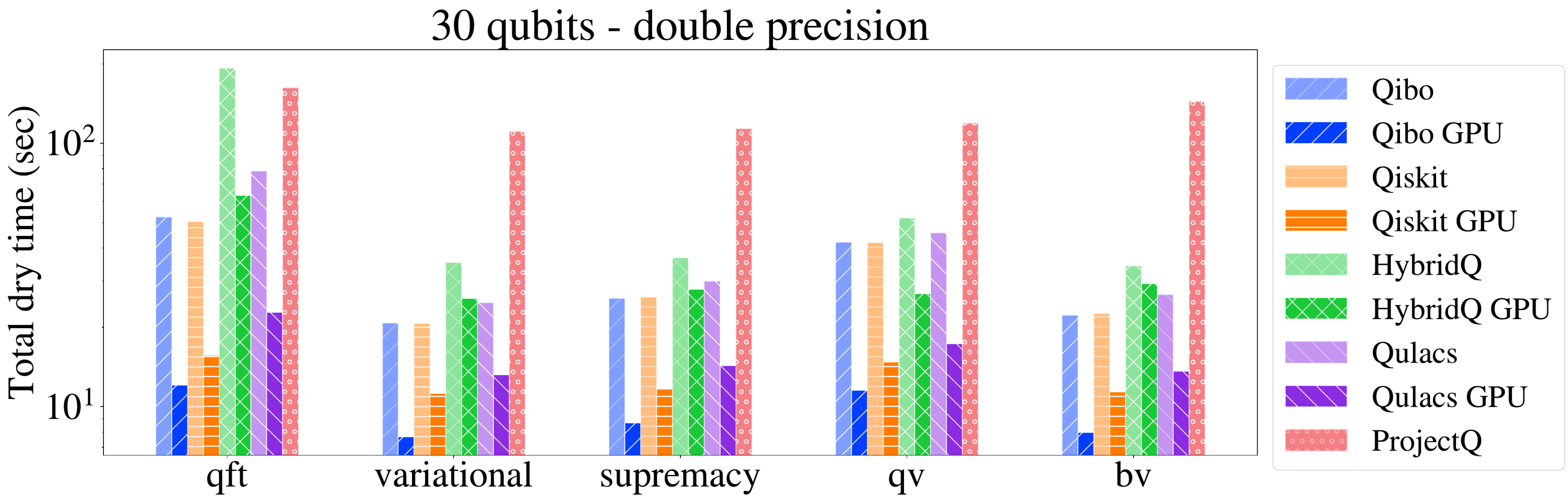
❌ Long computational times using naive approach (`Numpy` or `TensorFlow`) for circuits with large number of qubits.

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



2203.08826

Benchmark



All the benchmarks are available in qibojit-benchmarks.

Qibolab

Automatic deployment of quantum circuits on quantum hardware

Motivation

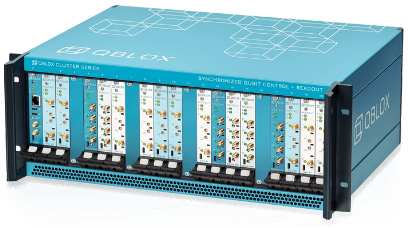
How are gates implemented on a quantum computer?

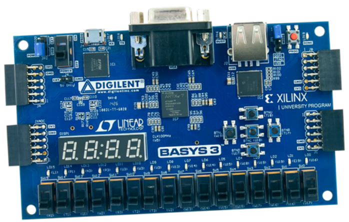
By sending microwave pulses.

How do we control them?

Using FPGAs

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





Pulse API example

```python
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
- FPGAs (based on Qick)

We also support local oscillators

- RohdeSchwarz SGS100A
- ERASynth

Introducing Qibocal

A reporting tool for calibration using Qibo

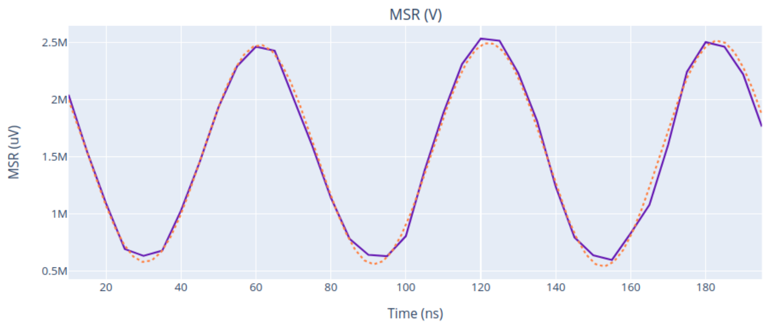
Motivation

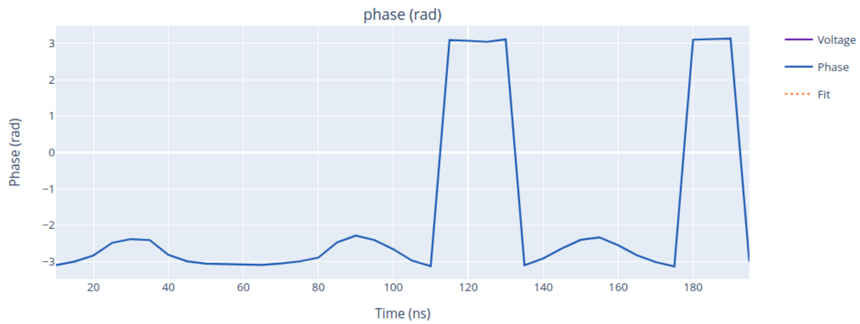
Let's suppose the following:

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

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

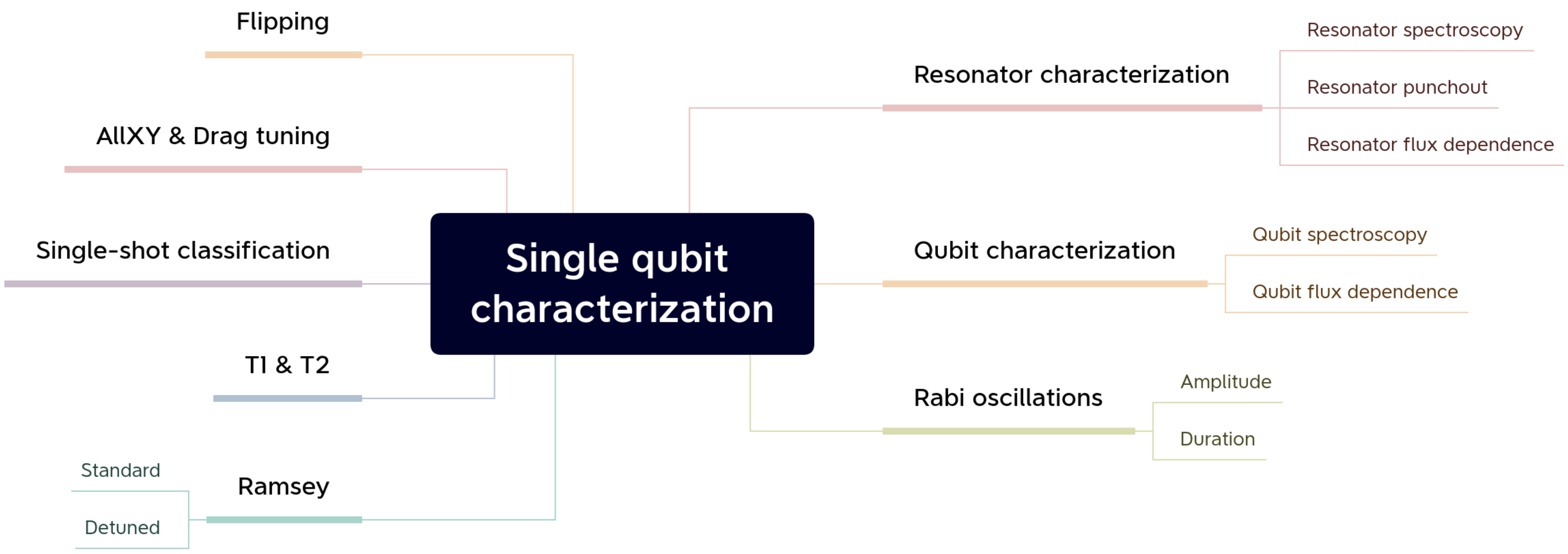
Characterization and **calibration** are an essential step to properly operate emerging quantum devices.



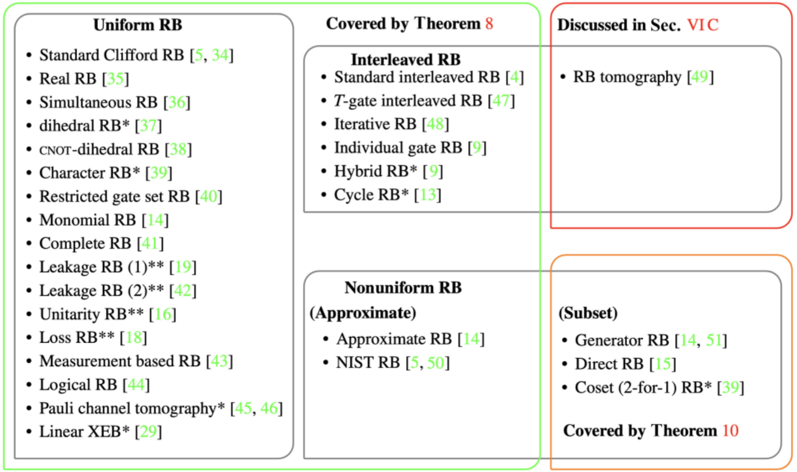


Calibration of RX pulse amplitude through a Rabi experiment through Qibocal.

Single Qubit Characterization: Pulse Level



Single Qubit Characterization: Circuit Level

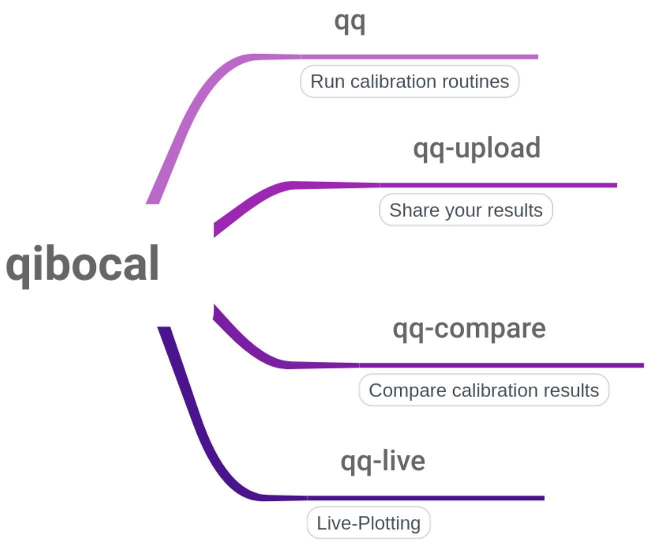


Currently in Qibocal the following protocols are implemented:

- 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



2303.10397

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

Qibocal Reports

- Home
  - Timestamp
  - Summary
- Actions
  - Ramsey - 0

Ramsey Experiment

Export to pdf

Platform: tii_rfsoc4x2
Run date: 2023-05-07
Start time (UTC): 05:20:51
End time (UTC): 05:21:19

Summary

In the table below we show the libraries and respective versions used in Ramsey Experiment.

Library	Version
numpy	1.23.5
qibo	0.1.13
qibocal	0.0.2
qibolab	0.0.3

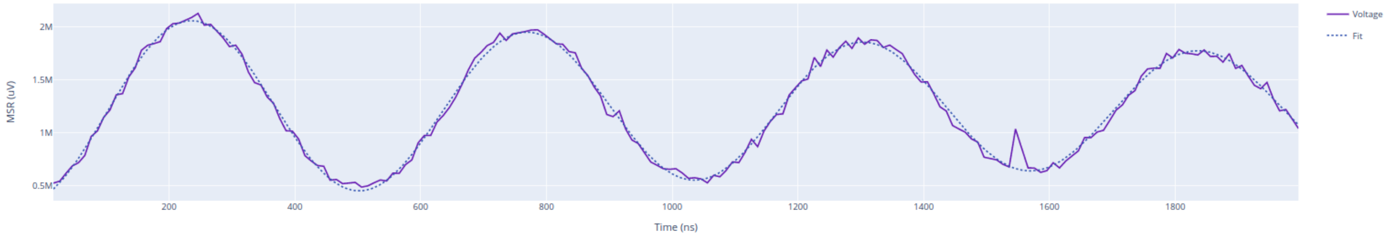
Actions

Please find below data generated by actions:

Ramsey - 0

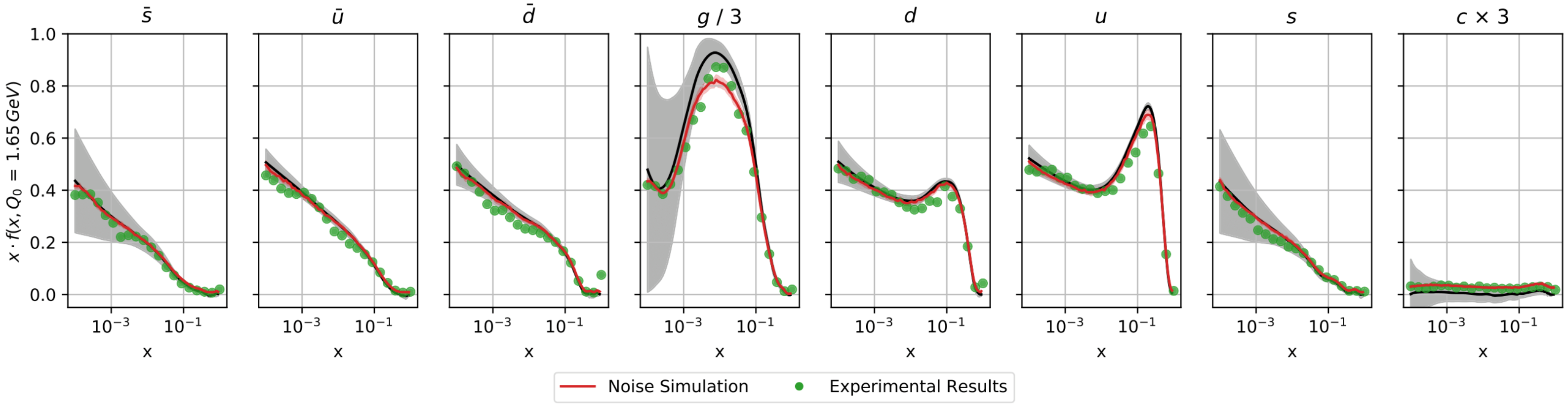
- Qubit 0

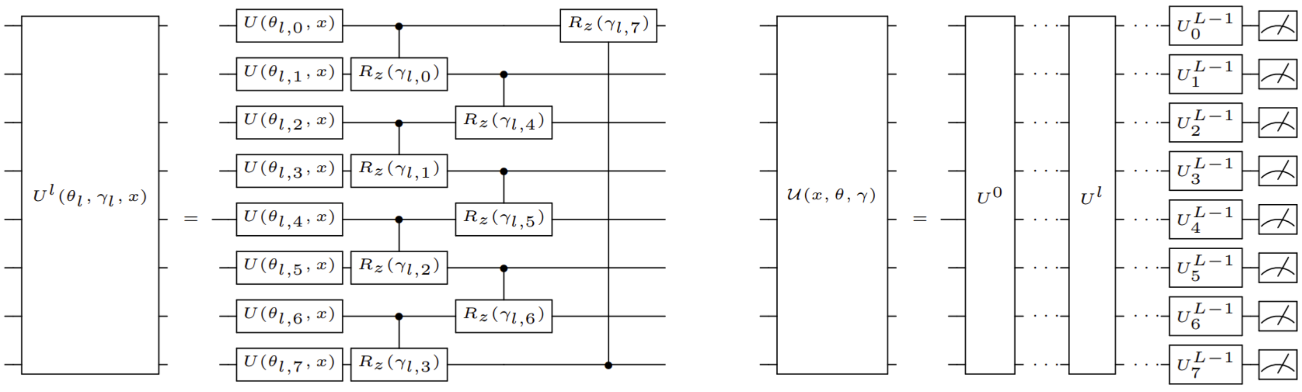
qubit	Fitting Parameter	Value
0	delta_frequency	-625,626.0 Hz
0	drive_frequency	5542303347.0 Hz



Applications

Determination of parton distribution functions using QML 2011.13934

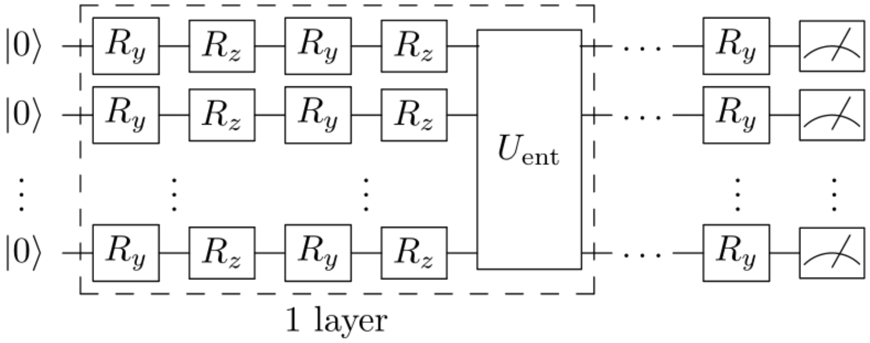




(a) One layer

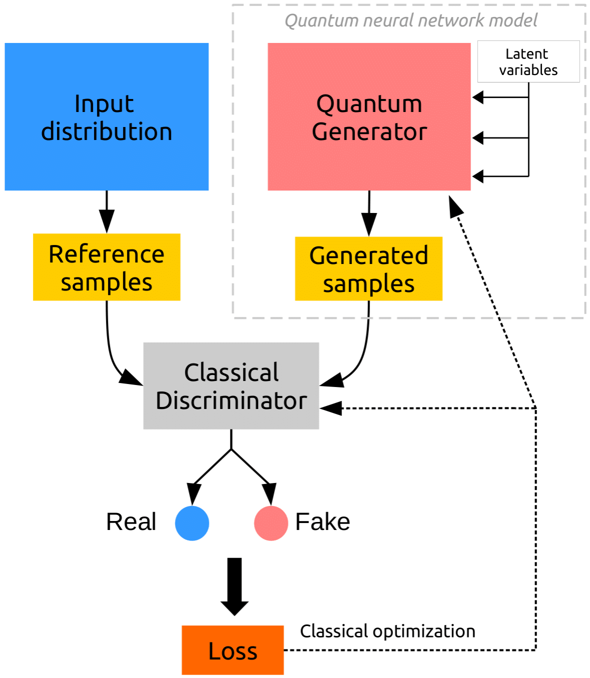
(b) Full Ansatz

MonteCarlo event generator using QGAN



Style-based approach

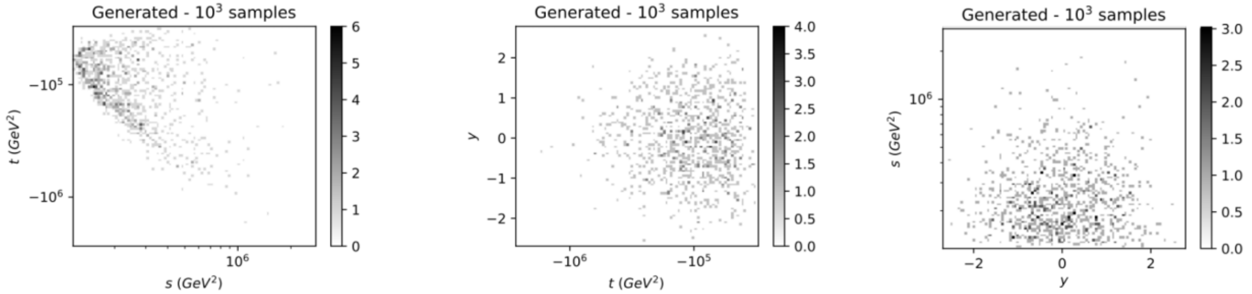
$$R_{y,z}^{l,m}\left(\boldsymbol{\phi_g},\boldsymbol{z}\right)=R_{y,z}\left(\phi_g^{(l)}z^{(m)}+\phi_g^{(l-1)}\right)$$



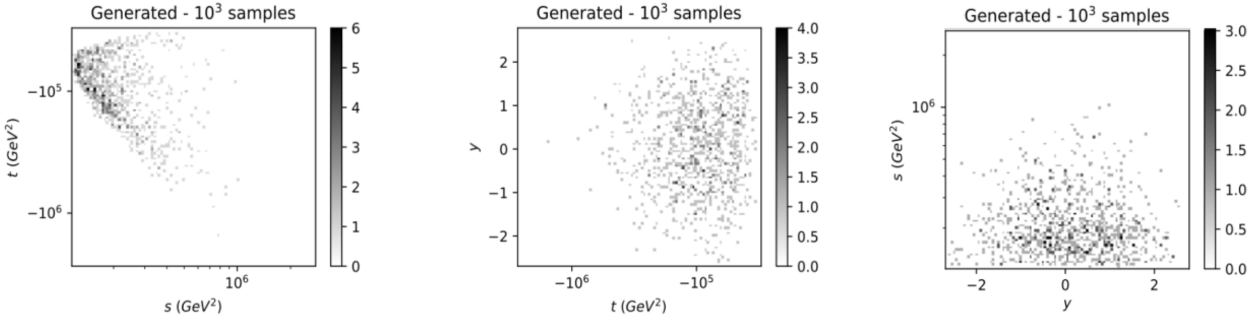
2110.06933

Results with $pp \to t\bar{t}$

ionQ samples:



IBM Q samples:



Implementation largely hardware independent!

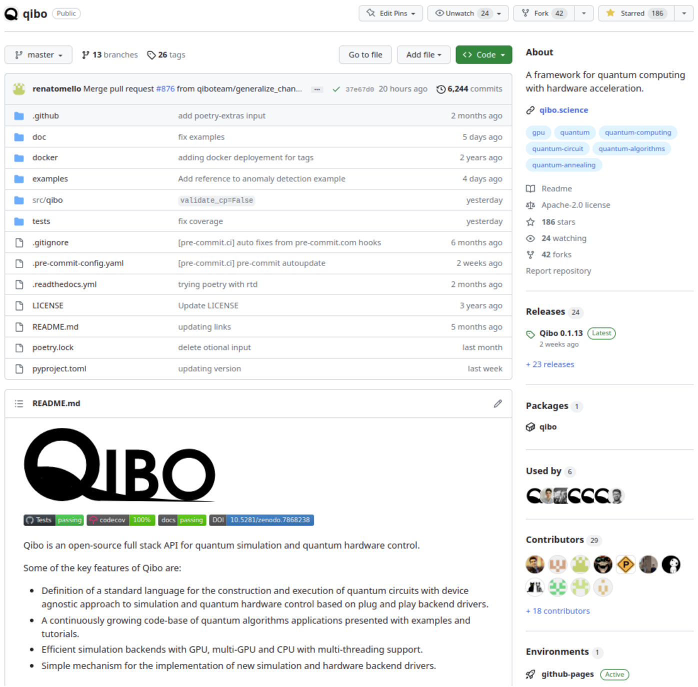
Outlook

We have presented Qibo, a simple full stack API capable of doing

- 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
- Many researchers are using it for HEP applications

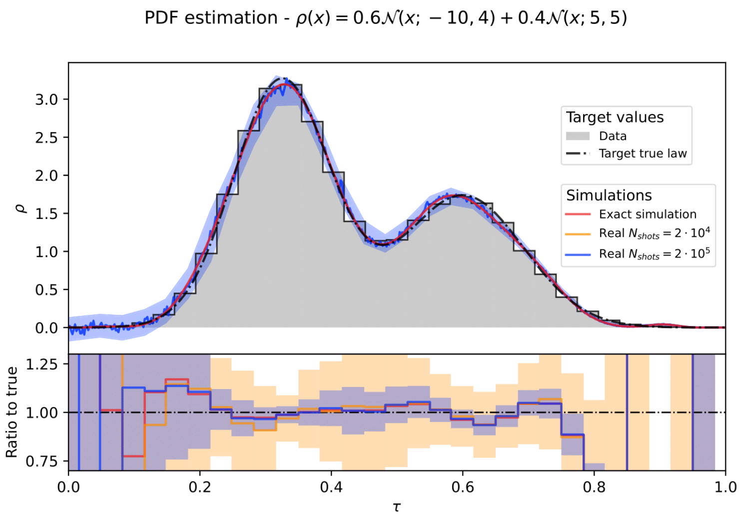


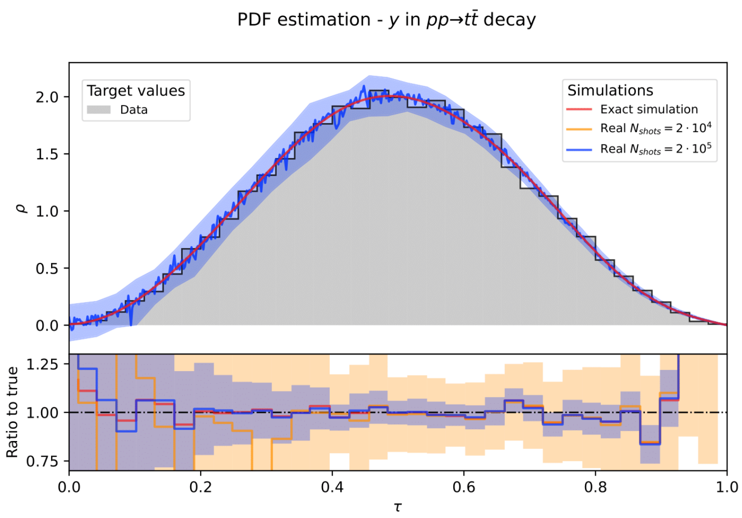
Thanks for listening!

Backup slides

FITTING PDFS USING ADIABATIC EVOLUTION

Use quantum adiabatic machine learning for the determination of PDF and sampling.

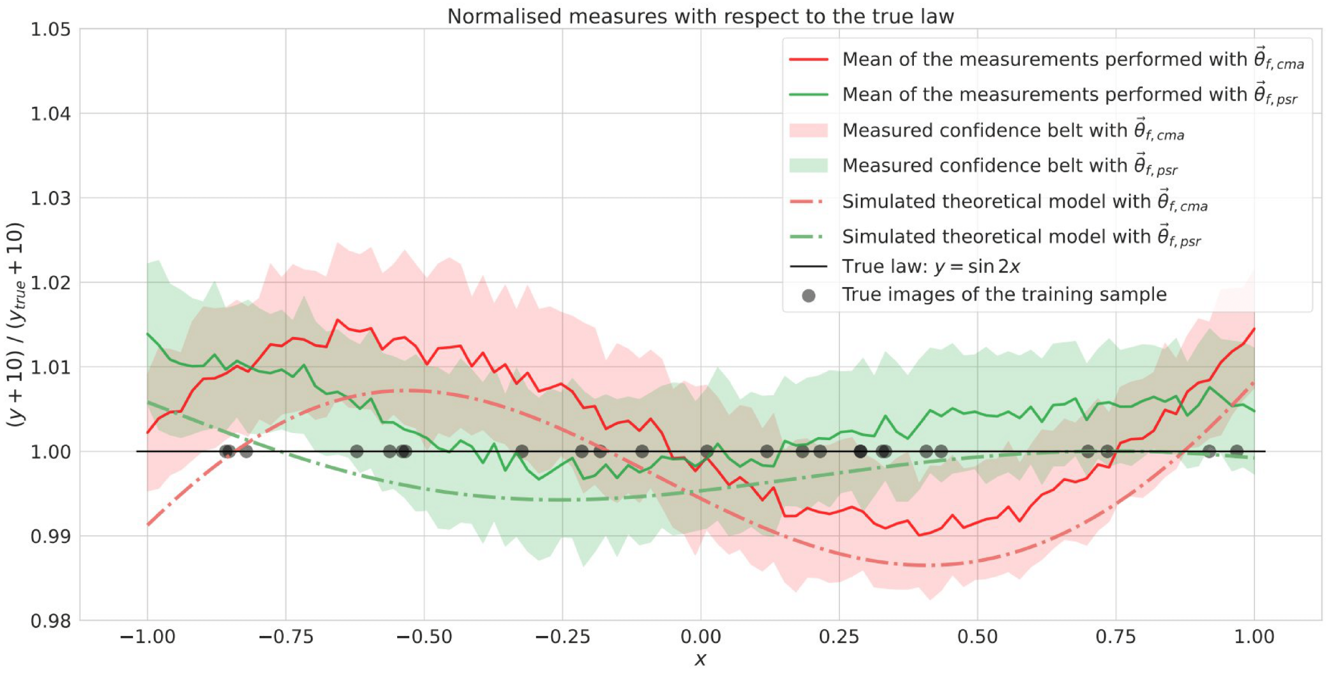




2303.11346

Paramater Shift Rule on Hardware

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



2210.10787