

# 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?

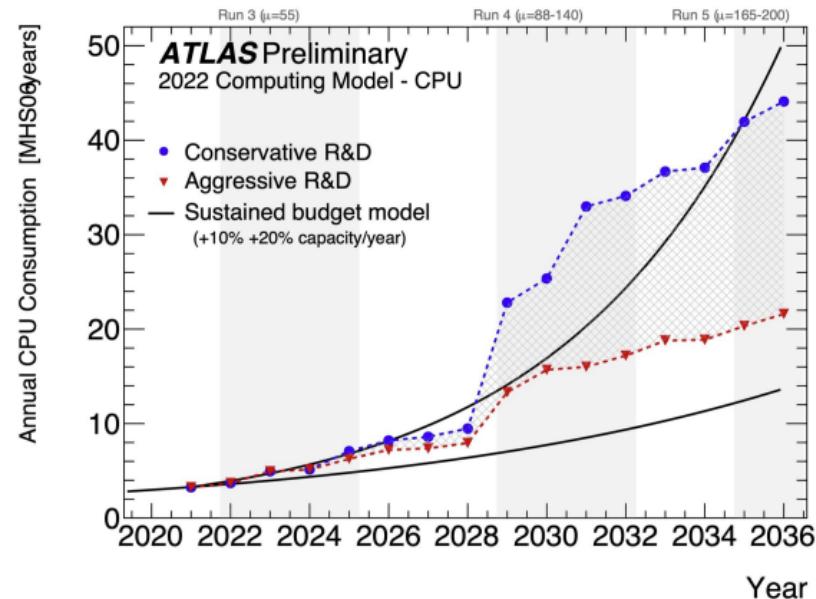
LHC produces TB of data per second which need to be processed.

Currently we **cannot** keep up with the computational resources required!

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

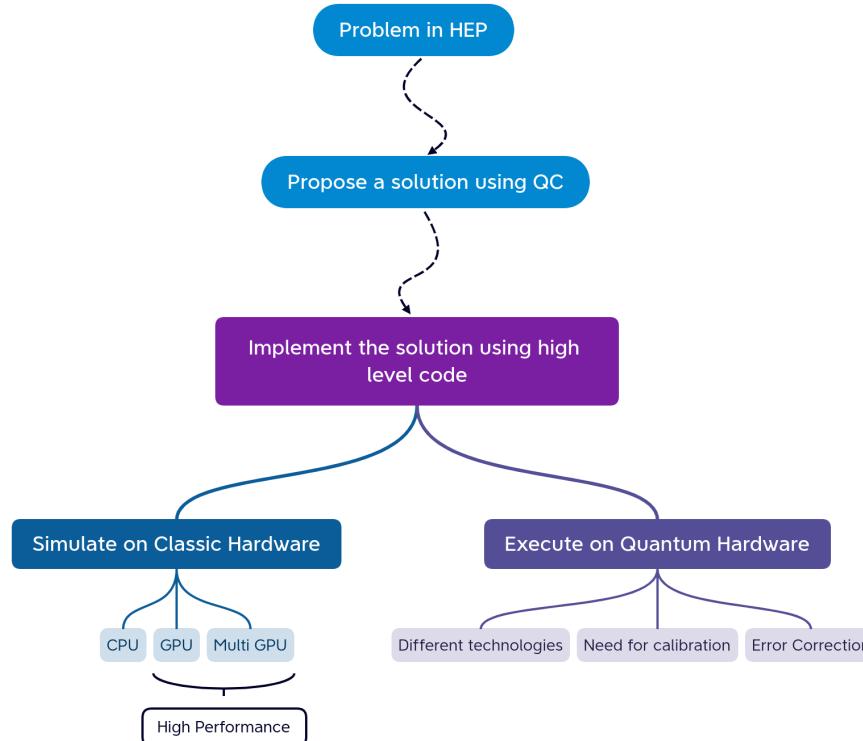
- Efficient in **high-dimensional** quantum state spaces
- Potential computational speed-ups
- Qubit representation offers **better compression**
- Acceleration of **classically expensive** operations (eigensolvers).

Nat Rev Phys 4, 143–144



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

# How CAN WE START USING QUANTUM COMPUTING?

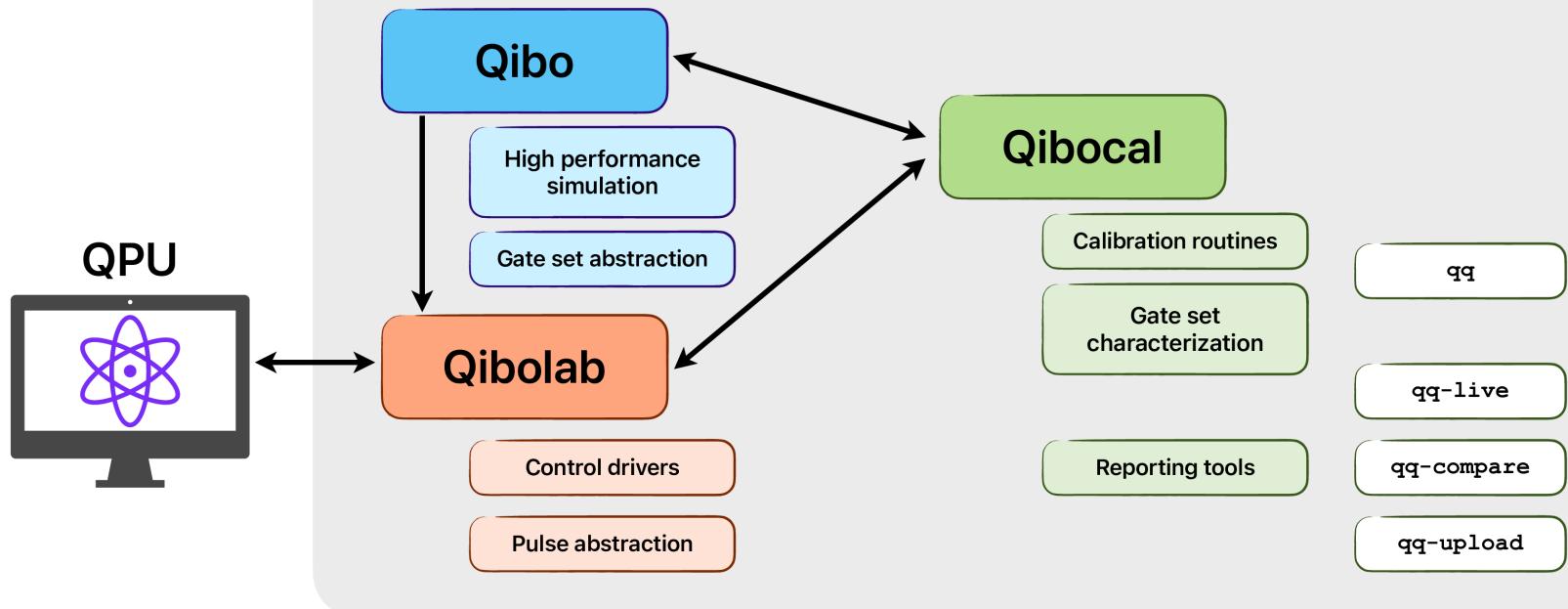


*Is it possible to create from scratch a framework for all of this?*

# INTRODUCING QIBO

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

# Qibo framework



# 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)
```

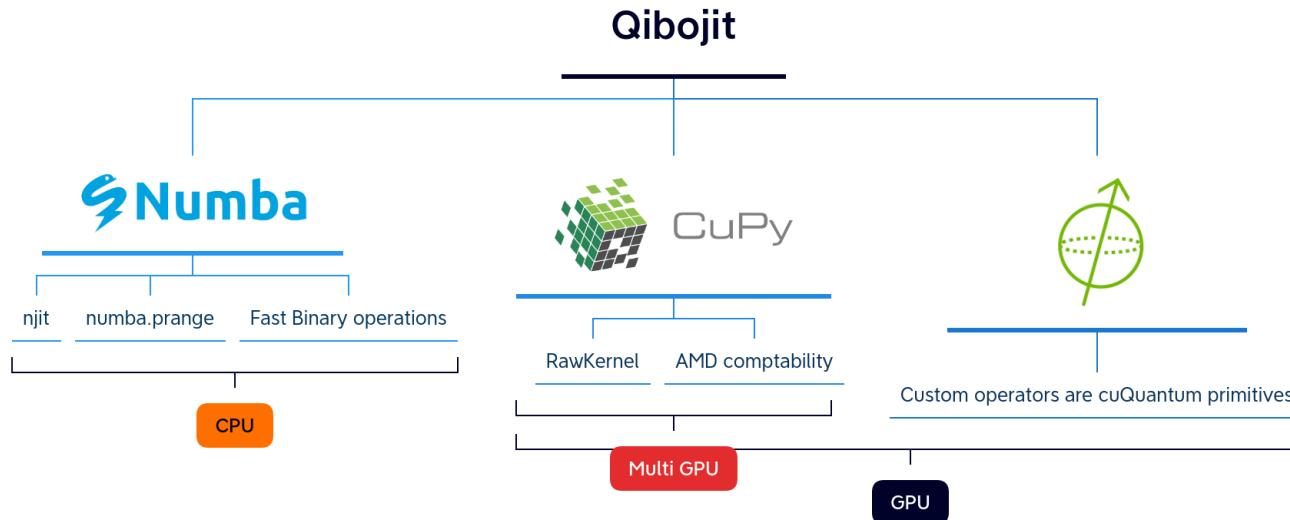
## QIBO FEATURES

- 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 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

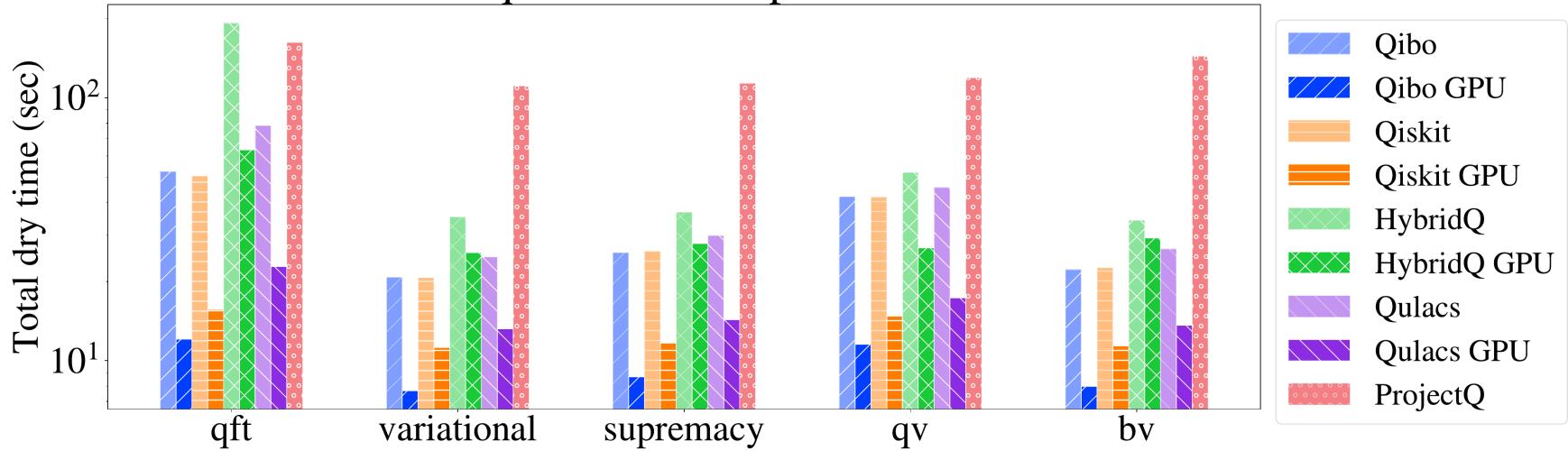
- ✗ Long computational times using naive approach (Numpy or TensorFlow) for circuits with large number of qubits.
- ✓ We need more sophisticated tools to be able to simulate a quantum circuits with more qubits!



2203.08826

# BENCHMARK

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
- FPGAs (based on Qick)

We also support local oscillators

- RohdeSchwarz SGS100A
- ERASynth

# INTRODUCING QIBOCAL

A reporting tool for calibration using Qibo

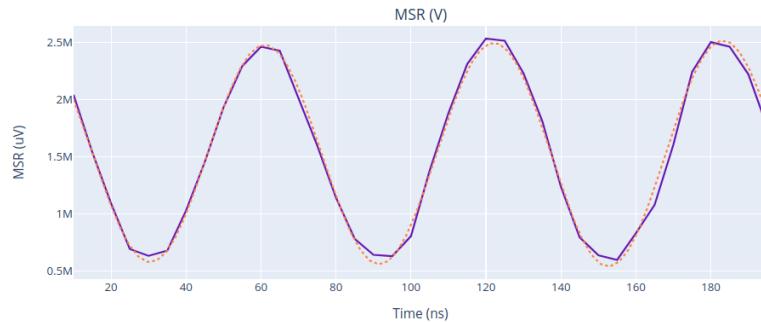
# MOTIVATION

Let's suppose the following:

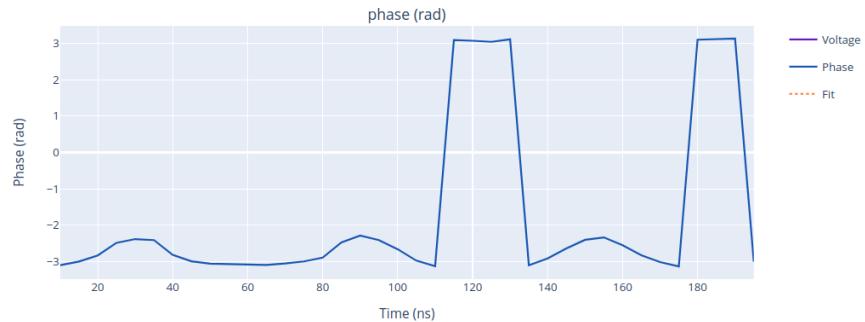
1. We have a QPU (self-hosted).
2. We have control over what we send to the QPU.
3. We know how to convert quantum circuits to pulses.

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

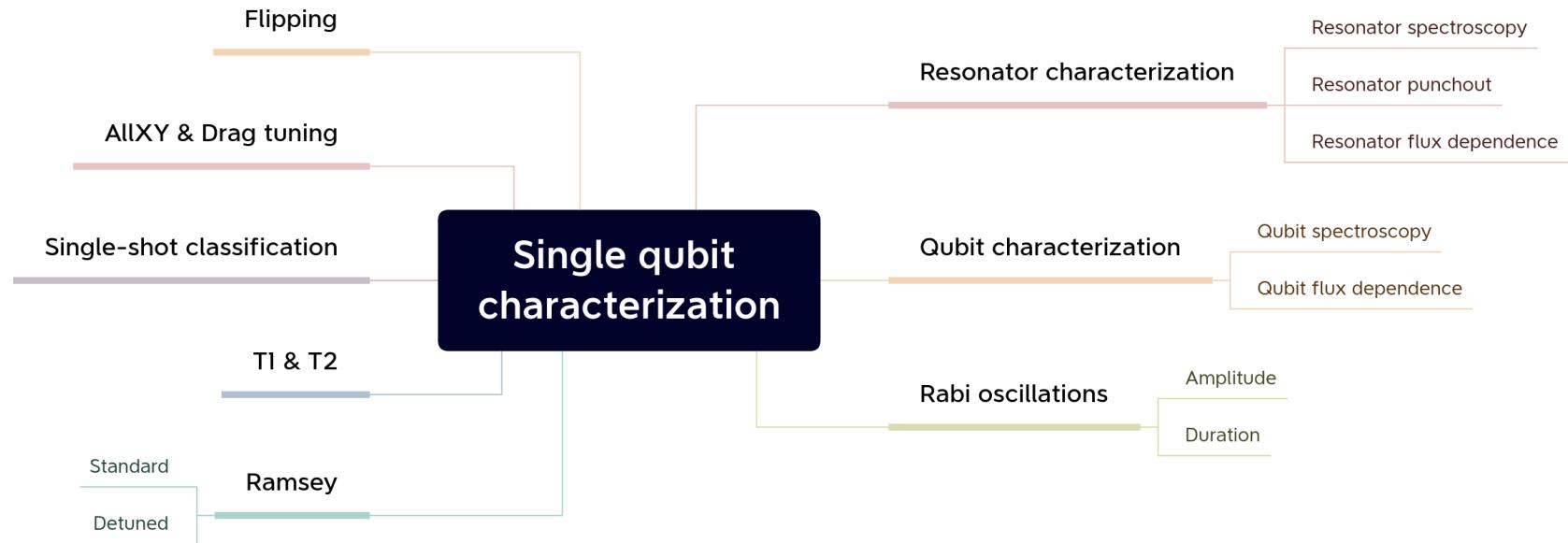
**Characterization** and **calibration** are an essential step to properly operate emerging quantum devices.



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



# SINGLE QUBIT CHARACTERIZATION: PULSE LEVEL



# SINGLE QUBIT CHARACTERIZATION: CIRCUIT LEVEL

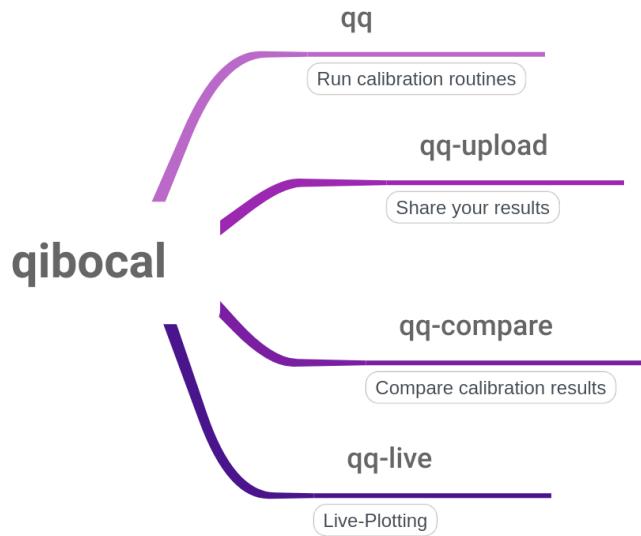
Uniform RB	Covered by Theorem 8	Discussed in Sec. VI C
<ul style="list-style-type: none"><li>• Standard Clifford RB [5, 34]</li><li>• Real RB [35]</li><li>• Simultaneous RB [36]</li><li>• dihedral RB* [37]</li><li>• CNOT-dihedral RB [38]</li><li>• Character RB* [39]</li><li>• Restricted gate set RB [40]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



2303.10397

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

# Ramsey Experiment

[Export to pdf](#)

Platform: tii\_rfsooc4x2  
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

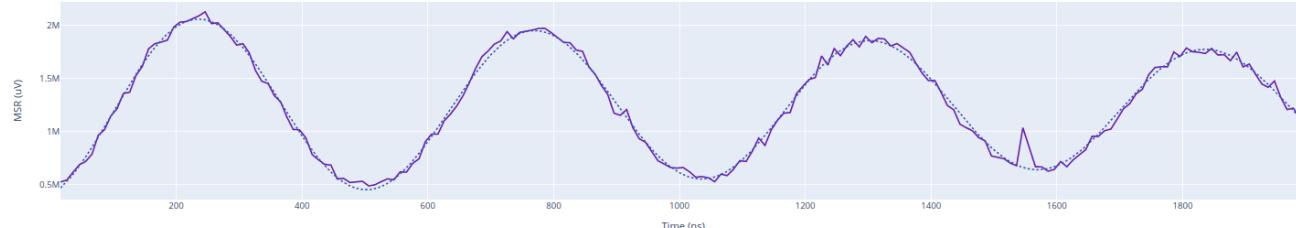
## Actions

Please find below data generated by actions:

### Ramsey - 0

#### - Qubit 0

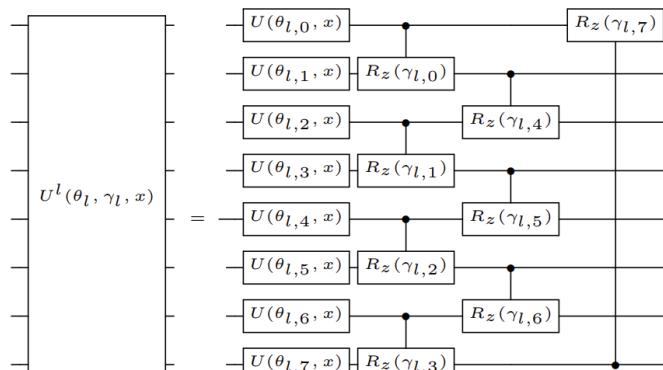
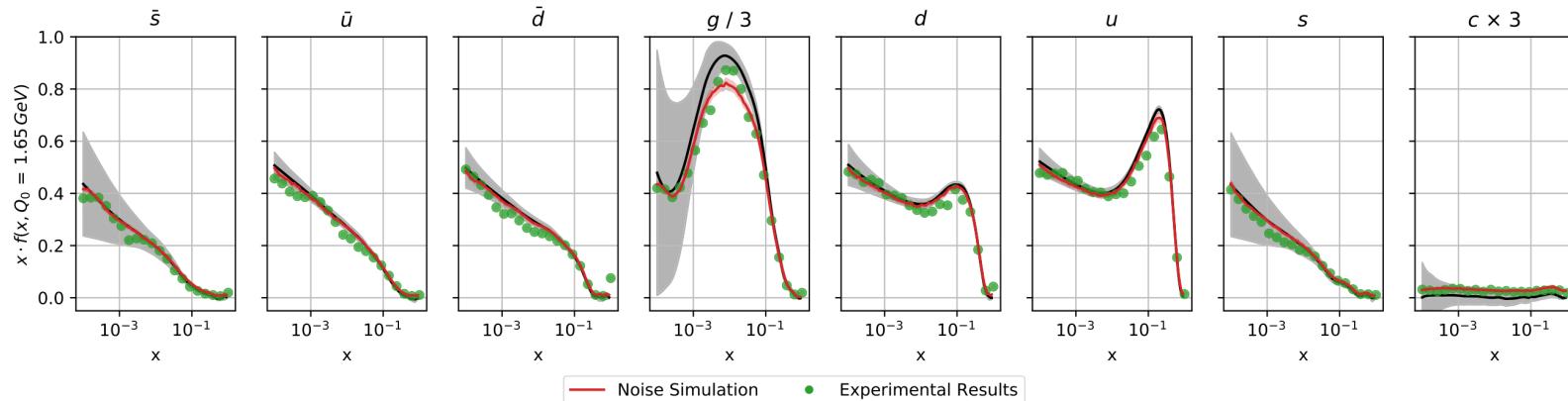
qubit	Fitting Parameter	Value
0	delta_frequency	-625,626.0 Hz
0	drive_frequency	5542303347.0 Hz



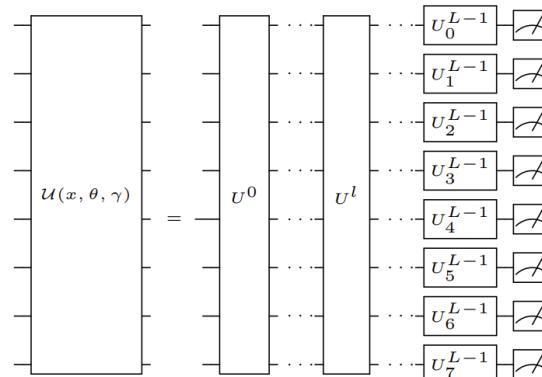
Voltage  
Fit

# APPLICATIONS

# DETERMINATION OF PARTON DISTRIBUTION FUNCTIONS USING QML 2011.13934

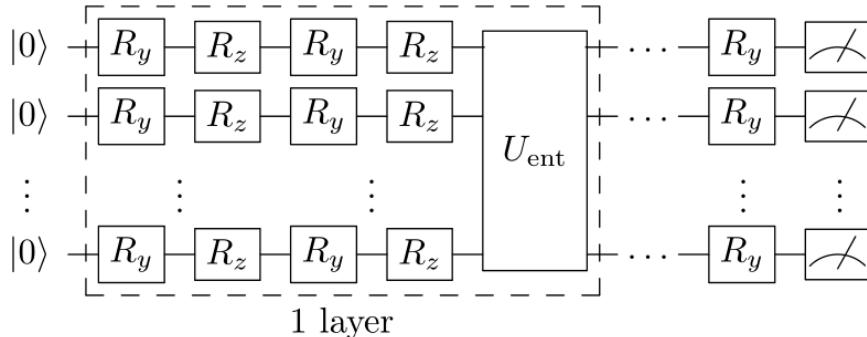


(a) One layer



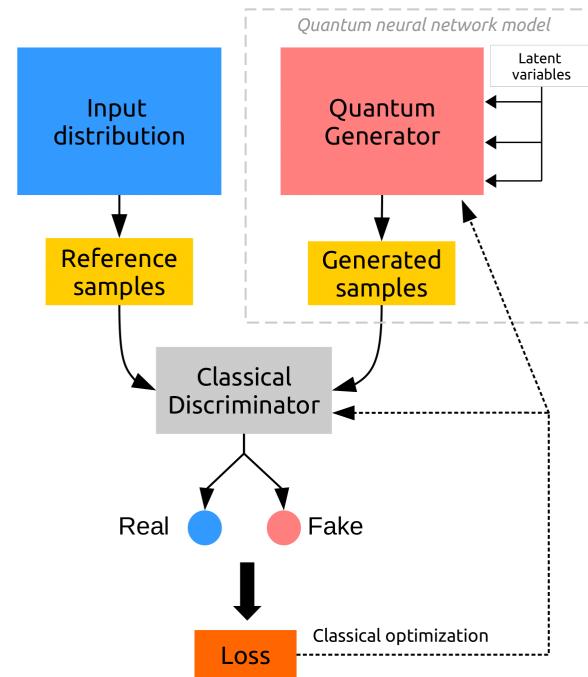
(b) Full Ansatz

## MONTECARLO EVENT GENERATOR USING QGAN



*Style-based approach*

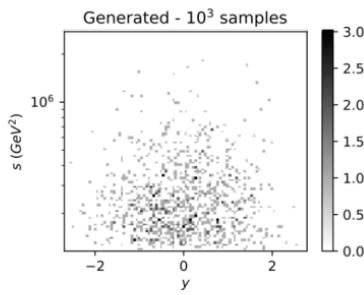
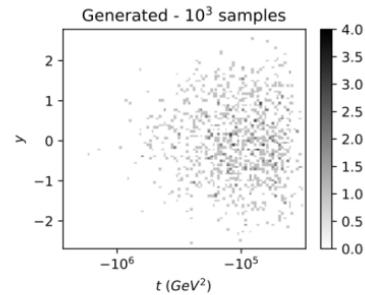
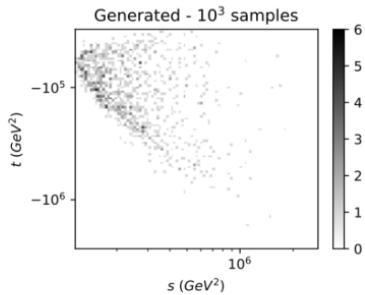
$$R_{y,z}^{l,m}(\phi_g, z) = R_{y,z} \left( \phi_g^{(l)} z^{(m)} + \phi_g^{(l-1)} \right)$$



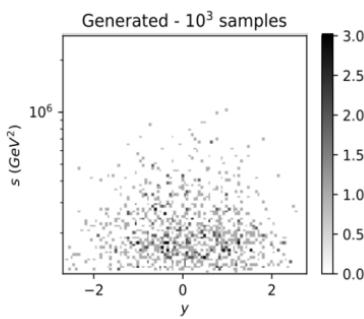
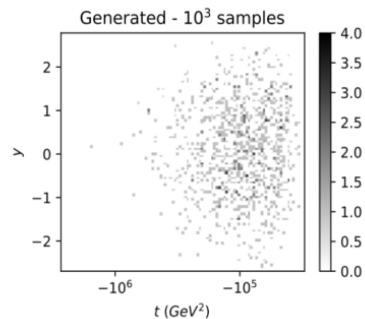
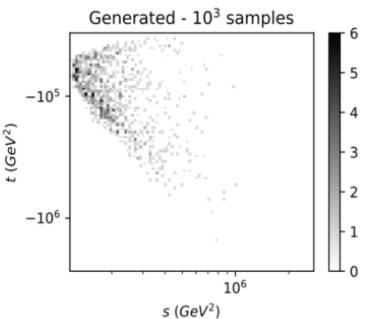
2110.06933

# RESULTS WITH $pp \rightarrow t\bar{t}$

**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

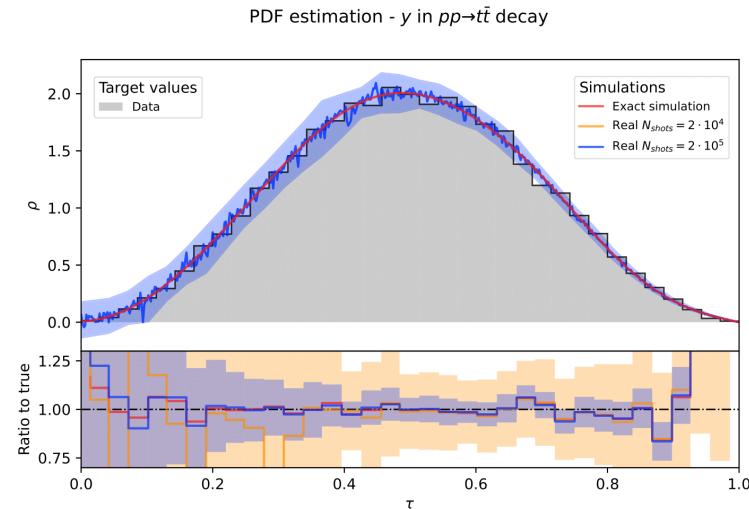
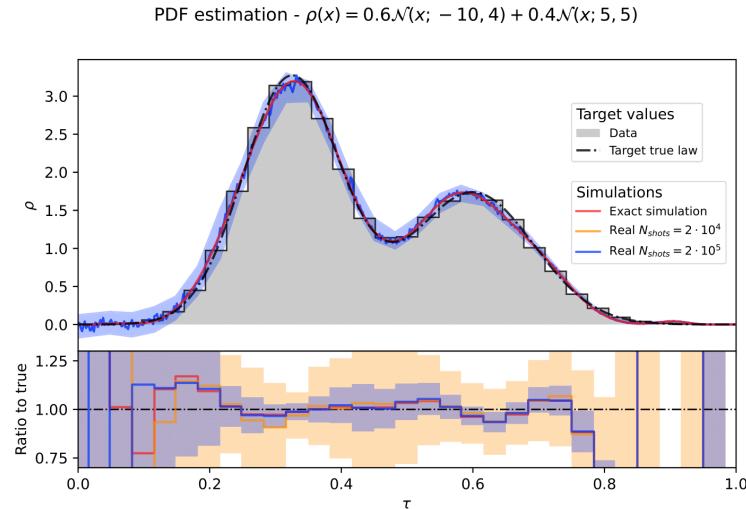
The screenshot shows the GitHub repository page for Qibo. At the top, there are buttons for 'Edit Pins', 'Unwatch', 'Fork', 'Starred', and 'Watch'. Below the header, the repository name 'qibo' and its status as 'Public' are displayed. It shows 1 master branch, 13 branches, and 26 tags. A 'Code' dropdown menu is open. The main area displays a list of recent commits from 'renatomello' with details like commit message, date, and author. To the right, there's a sidebar with sections for 'About', 'Tags', 'Readme', 'Licenses', 'Statistics', 'Releases', 'Packages', 'Used by', 'Contributors', and 'Environments'. The 'About' section describes Qibo as a framework for quantum computing with hardware acceleration. The 'Tags' section lists 'gpu', 'quantum', 'quantum-computing', 'quantum-circuit', 'quantum-algorithms', and 'quantum-annealing'. The 'Readme' section contains a large 'QIBO' logo. The 'Licenses' section shows an Apache-2.0 license. The 'Statistics' section includes metrics like 186 stars, 24 watching, and 42 forks. The 'Releases' section shows 'Qibo 0.1.13' as the latest release, updated 2 weeks ago. The 'Packages' section shows 'qibo' as a package. The 'Used by' section shows icons for various projects. The 'Contributors' section shows 18 contributors. The 'Environments' section shows 'github-pages' as active.

THANKS FOR LISTENING!

## BACKUP SLIDES

# FITTING PDFS USING ADIABATIC EVOLUTION

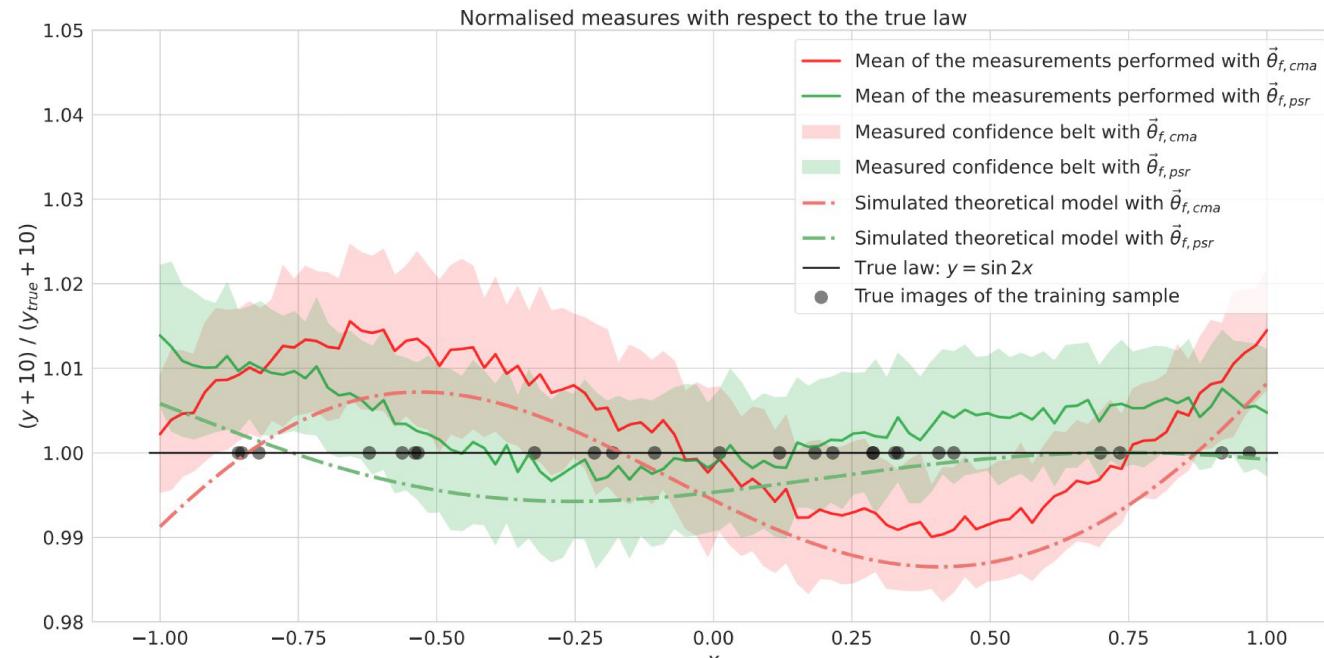
Use quantum adiabatic machine learning for the determination of PDF and sampling.



2303.11346

# PARAMATER SHIFT RULE ON HARDWARE

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



Scansionato con CamScanner

2210.10787