Introducing Qibo

Quantum simulation, control and calibration

Stefano Carrazza, on behalf of the Qibo team

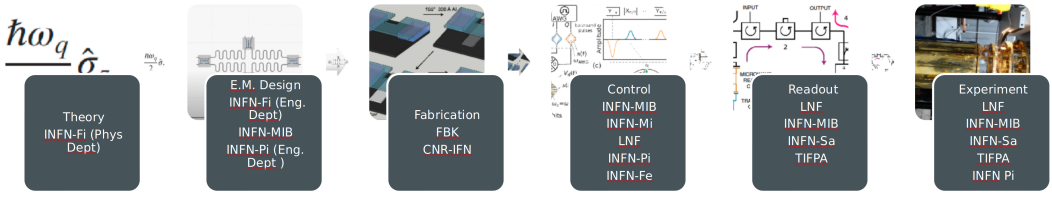
December 16th, 2022

SQMS, Padova

Introduction

Qub-IT project - CSN V

Qub-IT: Realization of an itinerant single-photon counter that surpasses present devices in terms of efficiency and low dark-count rates by exploiting repeated QND measurements of a single photon and entanglement in multiple qubits.

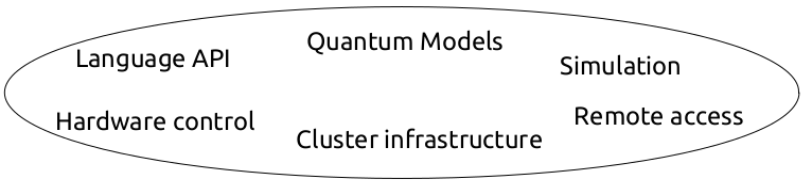


- Design and simulation of a SC qubit coupled to resonators.
- Fabrication of circuits with SC qubit.
- Single-shot measurement of SC qubit with quantum amplifier.
- Software for simulation, control and calibration of SC qubit.
- Quantum sensing experiment with entangled qubits.

Quantum software challenges

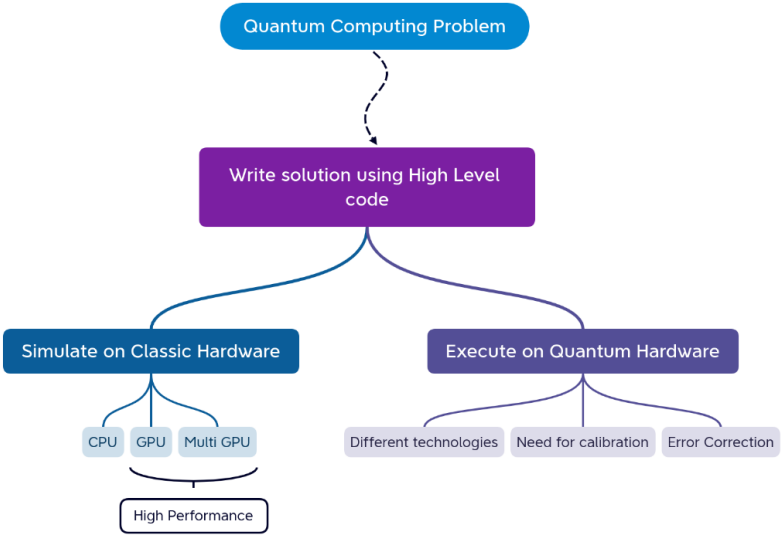
Researcher's needs (lab and theory):

1. Access to **interdisciplinary set of software tools** for:



2. **Open-source software** linked to benchmarks and publications.
3. Collaborative development and **definition of standards**.

Quantum software challenges



Cloud vs self-hosted

Cloud-based vs **self-hosted** devices and simulators:

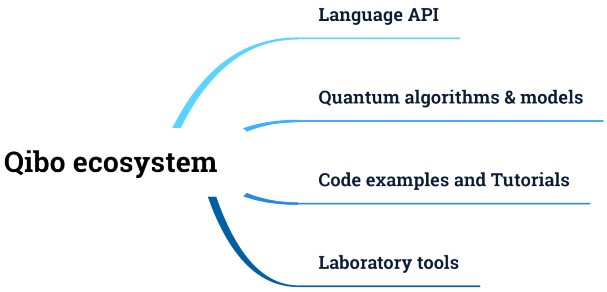
Industry (cloud-based)

Research labs and HPC (self-hosted)

Introducing Qibo

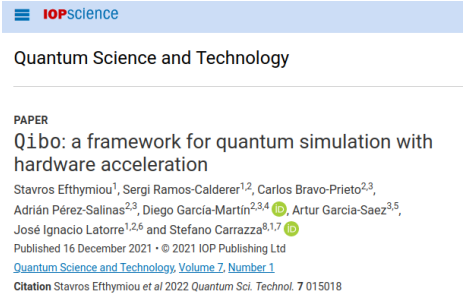
Introducing Qibo

Qibo is an open-source full stack **API** for quantum simulation and hardware control.
It is platform **agnostic** and supports **multiple backends**.



https://github.com/qiboteam/qibo

Introducing Qibo



IOPscience

Quantum Science and Technology

PAPER

Qibo: a framework for quantum simulation with hardware acceleration

Stavros Efthymiou[1], Sergi Ramos-Calderer[1,2], Carlos Bravo-Prieto[2,3], Adrián Pérez-Salinas[2,3], Diego García-Martín[2,3,4], Artur Garcia-Saez[3,5], José Ignacio Latorre[1,2,6] and Stefano Carrazza[8,1,7]

Published 16 December 2021 • © 2021 IOP Publishing Ltd

Quantum Science and Technology, Volume 7, Number 1

Citation Stavros Efthymiou *et al* 2022 *Quantum Sci. Technol.* **7** 015018

https://arxiv.org/abs/2009.01845

Quantum

the open journal for quantum science

Quantum simulation with just-in-time compilation

Stavros Efthymiou[1], Marco Lazzarin[1], Andrea Pasquale[1,2], and Stefano Carrazza[1,2,3]

[1]Quantum Research Centre, Technology Innovation Institute, Abu Dhabi, UAE.
[2]TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Milan, Italy.
[3]CERN, Theoretical Physics Department, CH-1211 Geneva 23, Switzerland.

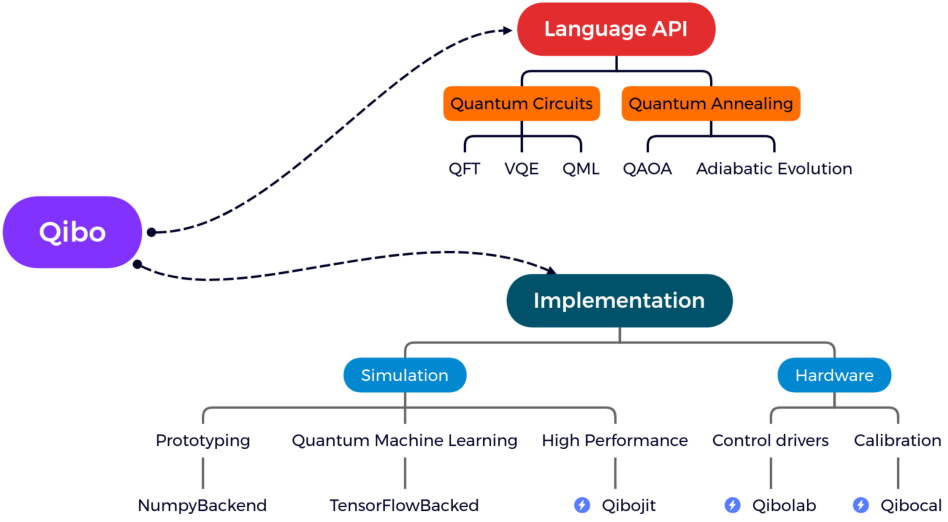
Published:	2022-09-22, volume 6, page 814
Eprint:	arXiv:2203.08826v2
Doi:	https://doi.org/10.22331/q-2022-09-22-814
Citation:	Quantum 6, 814 (2022).

https://arxiv.org/abs/2203.08826

Laboratory	Country	Technology	Qubits
INFN	Italy	Superconducting	1
TII	United Arab Emirates	Superconducting	1, 2, 5, 20
Qilimanjaro	Spain	Superconducting	1 and 2
CQT	Singapore	SC and trapped ion	up to 10

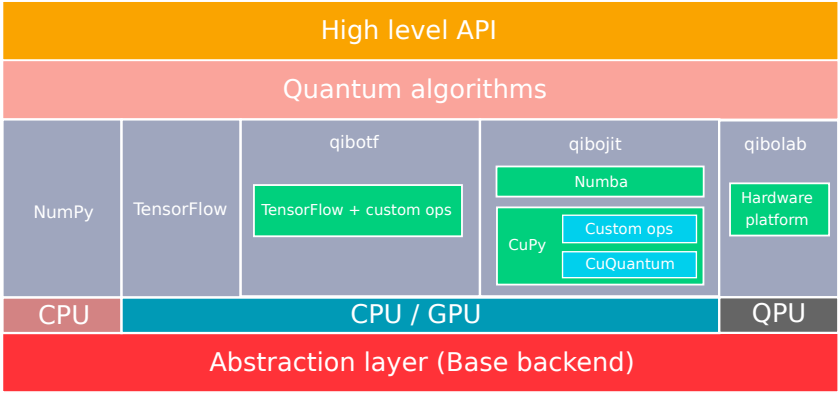
+70 collaborators

Backends in Qibo



Abstractions in Qibo

Qibo stack



Qibo simulation benchmarks

Introducing Qibojit

State vector simulation solves:

$$\psi'(\sigma_1, \ldots, \sigma_n) = \sum_{\boldsymbol{\tau}'} G(\boldsymbol{\tau}, \boldsymbol{\tau}')\psi(\sigma_1, \ldots, \boldsymbol{\tau}', \ldots, \sigma_n)$$

number of operations scales exponentially with the number of qubits. **Qibo** uses just-in-time technology:

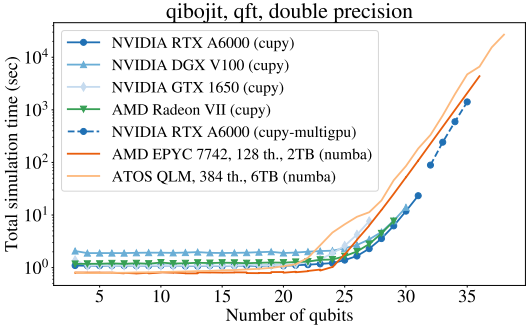
- CPU: Numpy tensor, custom operations with Numba.
- GPU(s): CuPy tensors, custom operations using:
  - CuPy JIT raw kernels
  - NVIDIA cuQuantum API

```python
from numba import njit, prange

@njit(parallel=True, cache=True)
def apply_gate_kernel(state, gate, target):
    """Operator that applies an arbitrary one-qubit gate.

    Args:
        state (np.ndarray): State vector of size (2 ** nqubits,).
        gate (np.ndarray): Gate matrix of size (2, 2).
        target (int): Index of the target qubit.
    """
    k = 1 << target
    # for one target qubit: loop over half states
    nstates = len(states) // 2
    for g in prange(nstates):
        # generate index with fast binary operations
        i1 = ((g >> m) << (m + 1)) + (g & (k - 1))
        i2 = i1 + k
        state[i1], state[i2] = (gate[0, 0] * state[i1] + \
                                gate[0, 1] * state[i2],
                                gate[1, 0] * state[i1] + \
                                gate[1, 1] * state[i2])
    return state
```

Qibojit features

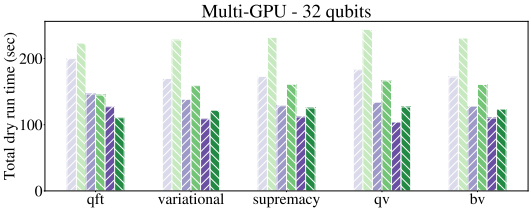
In-place updates, specialized operators for single and two qubit gates (exploit sparsity).

Qibojit



Qibojit

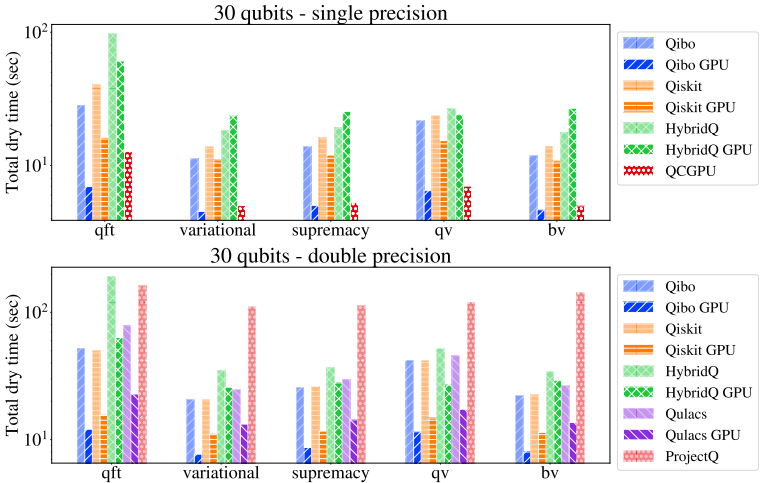
- Supports CPU, GPU and multi-GPU.
- NVIDIA and AMD GPUs.
- Reduced memory footprint.



Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks` [arXiv:2203.08826]

Qibo vs other libraries

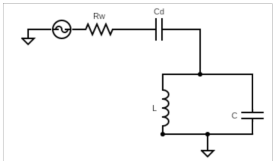
Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks` [arXiv:2203.08826]



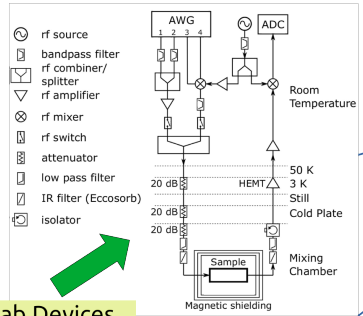
Quantum hardware control

Software control challenges

Transmon
(qubit)



Platform design



Lab Devices

Cryostat
(Fridge)

QRC@TII Labs

Software control challenges

Cryostat
(Fridge)

Modulated signal (pulse)

pulses

Input

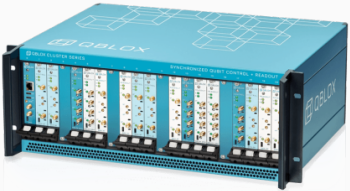
Readout

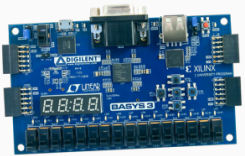
Microwave source
Local oscillators
Amplifiers
FPGAs
...

Experimental Devices

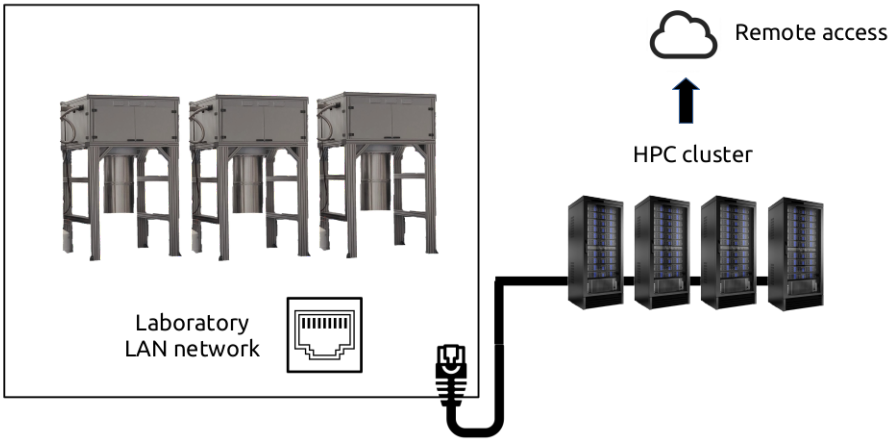
Laboratory
LAN network

Qibolab deployment

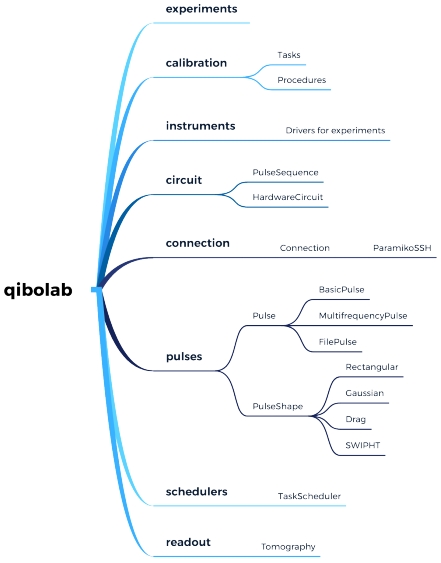




Software control challenges



Introducing qibolab

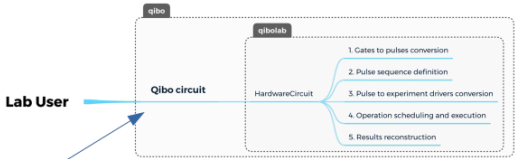


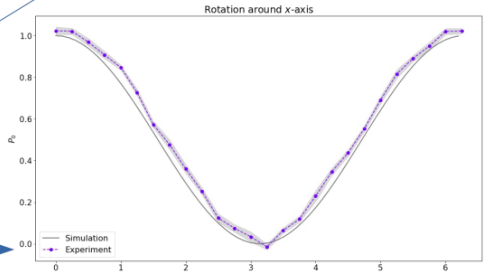
Qibolab key features:

- Create **custom experimental drivers** for lab setup.
- Platform **agnostic** layout.
- Deploy **Qibo models** on quantum hardware easily

Example

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

# select experimental backend
qibo.set_backend("qibolab", platform='tiiq')

# create circuit
circuit = models.Circuit(1)
circuit.add(gates.RX(0, theta=0))
circuit.add(gates.M(0))

exp_angles = np.arange(0, 2*np.pi, 0.25)
for t in exp_angles:
    # set RX rotation angle
    circuit.set_parameters([t])

    # execute the circuit
    state   = circuit.execute(nshots=1024)

    # calculate the probabilities of |0> and |1>
    probs_0, probs_1 = state.probabilities(qubits=(0,))
```





Supported instruments

- AWG:
  - Tektronix (e.g. AWG5204, AWG70000A)
  - AlazarTech boards (e.g. ATS9371)
  - QuickSyn
  - Rigol (DC 5072)
- QBlox cluster
- Zurich Instruments
- Quantum Machines
- FPGA boards:
  - QICK: RFSoC

Development roadmap:

- Qibo already provides a prototype approach based on AWG-like instruments.
- We are working on production control hardware based on FPGA boards.

Quantum calibration software

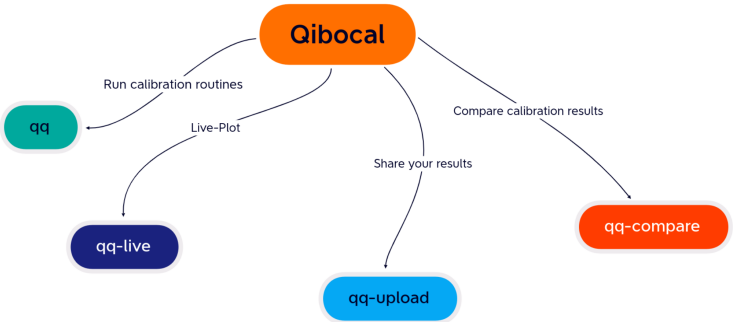
Introducing Qibocal

We are developing a new tool called Qibocal to perform qubits calibration in Qibo using Qibolab as the main driver.

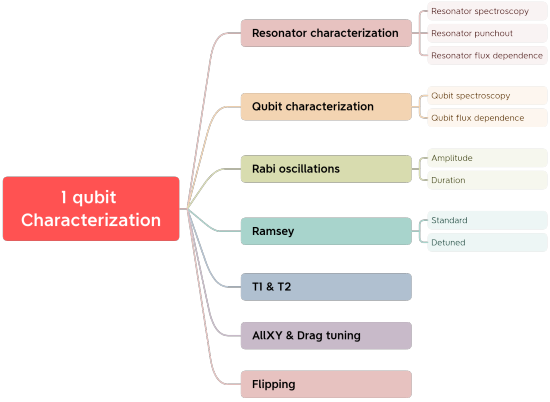
The main features are:

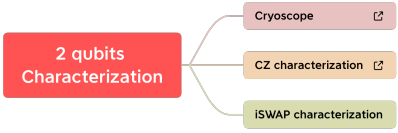
- Platform agnostic approach
- Launch calibration routines easily
- Live-plotting tools
- Live-fitting tools
- Save and share results
- Autocalibration

Qibocal: implementation

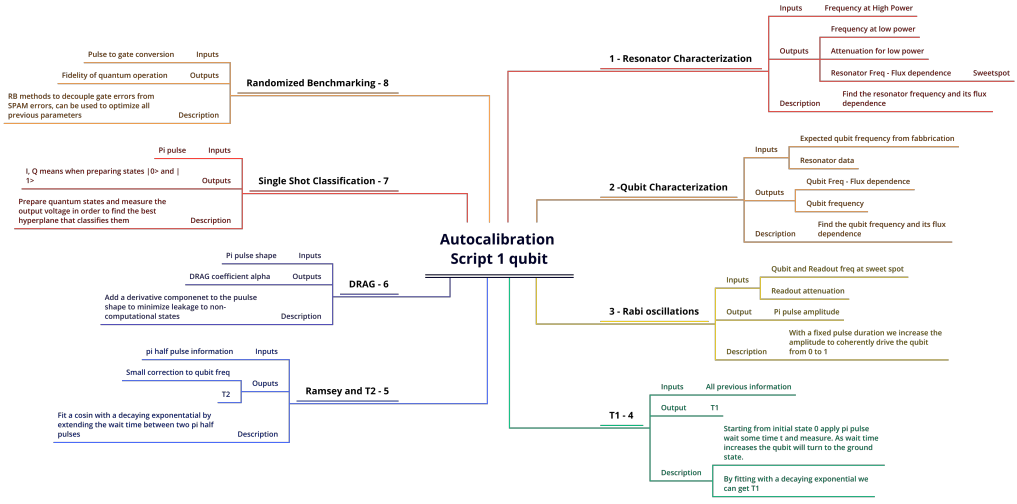


Introducing Qibocal





Introducing Qibocal



Live plotting and automatic calibration

Qibocal Reports

Home
Timestamp
Summary

Actions
Resonator Spectroscopy
MSR and Phase vs Frequency
Rabi Pulse Length
MSR vs Time

2022-10-18-000-andrea-pasquale

Platform: tiiq
Run date: 2022-10-18
Start time (UTC): 08:43:01
End time (UTC): 08:43:59

Summary

In the table below we show the libraries and respective versions used in 2022-10-18-000-andrea-pasquale.

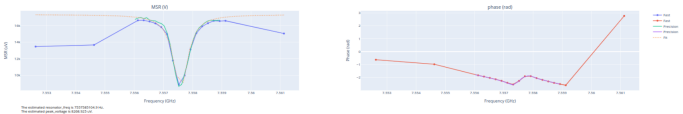
Library	Version
numpy	1.23.4
qibo	0.1.8
qibocal	0.0.1.dev0
qibolab	0.0.1.dev0

Actions

Please find below data generated by actions:

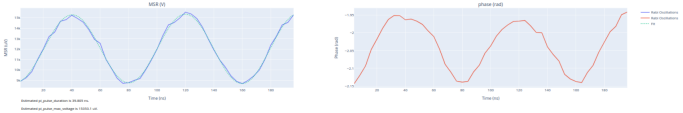
Resonator Spectroscopy

MSR and Phase vs Frequency - Qubit 2



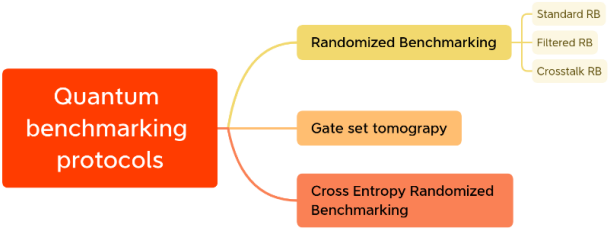
Rabi Pulse Length

MSR vs Time - Qubit 2



Quantum benchmarking protocols

Quantum hardware we need to compute the gates error behavior. In the current state-of-the-art this is computed using Quantum benchmarking protocols.

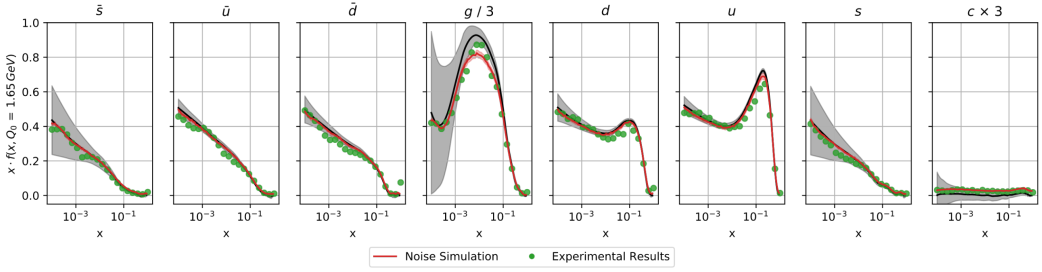


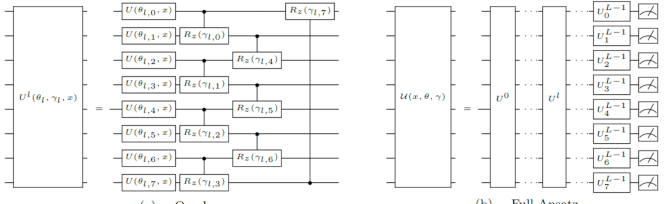
In Qibocal we are currently developing a suite for the execution of the latest QBP available.

Applications in HEP

Determination of parton distribution functions using QML

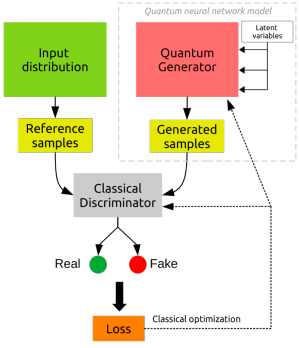
A. Salinas et al, **Determining the proton content with a quantum computer**, PRD, 2011.13934.

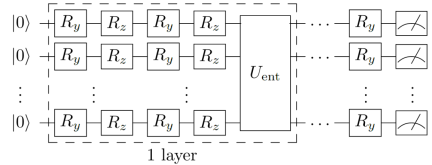


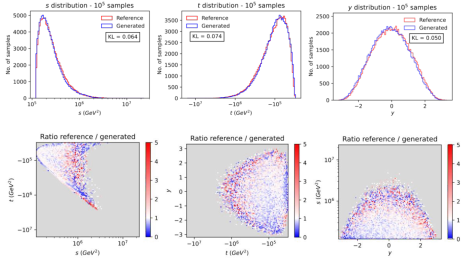


MC event generation using Style-qGAN

C. Bravo-Prieto et al, **Style-based quantum generative adversarial networks for Monte Carlo events**, Quantum, 2110.06933.

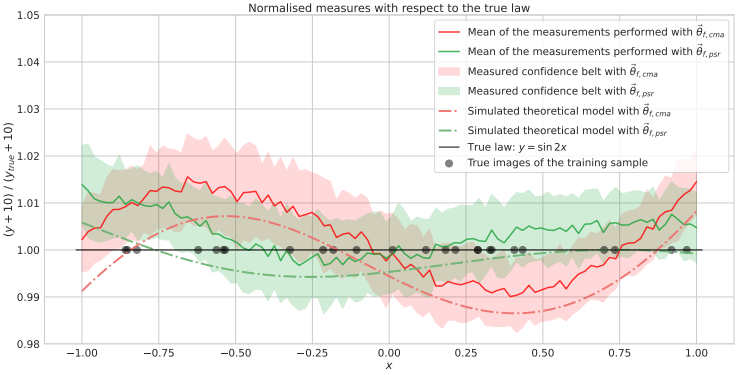






Gradient descent on a QPU

M. Robbiati et al, **A quantum analytical Adam descent through parameter shift rule using Qibo**, ICHEP 2022 proceedings, 2210.10787.



Outlook

Outlook

Qibo accommodates different tasks:

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

Unique features in Qibo:

1. All modules are open source.
2. Modular layout design with possibility of adding
   - new backends for simulation
   - new platforms for hardware control
3. Community driven effort

`https://github.com/qiboteam/qibo`

`https://qibo.readthedocs.io`