A Photonic Quantum Computer Simulation Software Platform

Zoltán Kolarovszki[1, 2], Péter Rakyta[1, 2], Ágoston Kaposi[1, 2], Boldizsár Poór[3], Szabolcs Jóczik[1], Tamás Kozsik[2], and Zoltán Zimborás[1, 4]

[1]Wigner Research Centre for Physics, [2]Eötvös Loránd University, [3]University of Oxford, [4]Algorithmiq

www.piquasso.com piquassoquantum@gmail.com Budapest-Quantum-Computing-Group

We introduce the Piquasso quantum programming framework, a full-stack open-source platform for the simulation and programming of photonic quantum computers. Piquasso can be programmed via a high-level Python programming interface enabling users to perform efficient quantum computing with discrete and continuous variables. Via optional high-performance C++ backends Piquasso provides state-of-the-art performance in the simulation of photonic quantum computers. The Piquasso framework is supported by an intuitive web-based graphical user interface where the users can design quantum circuits, run computations and visualize the results.

Piquasso Python library

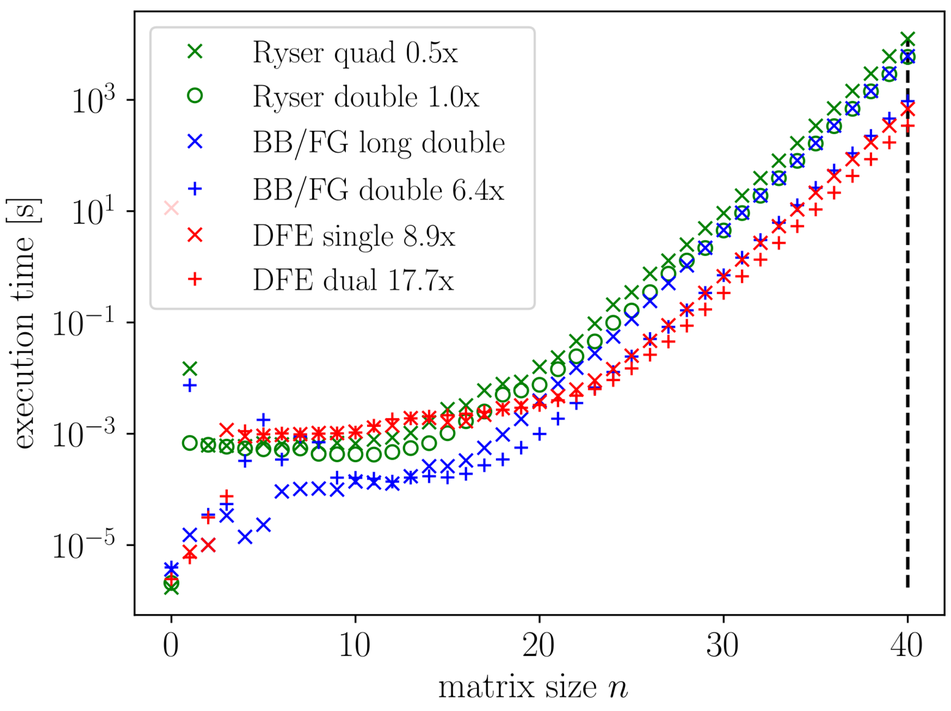
Piquasso implements a different set of calculations for different computational models corresponding to certain special cases, such as computations solely with pure states or Gaussian states. The explicit use of these special cases results in certain benefits, e.g. using Gaussian states are generally faster than using Fock states.

On the right is a basic Piquasso example. On line 5, the program definition starts with a `with` statement that will result in a `Program` instance. In this block, all the instructions should be specified. The `pq.Q` class is used to specify the modes on which the instruction on the other side of the `|` or `__or__` operator. After the program definition, a `GaussianSimulator` instance is instantiated with two modes, and the result is acquired by executing `program` with `simulator`. The execution parameter `shots=100` is specified to acquire 100 samples from the measurement.

Example code simulating Gaussian circuits

```
1  import numpy as np
2  import piquasso as pq
3
4  # Beginning of program definition
5  with pq.Program() as program:
6      # Quantum Gates
7      pq.Q(0, 1) | pq.Displacement(alpha=1.0)
8      pq.Q(0)    | pq.Squeezing(r=0.1, phi=np.pi / 3)
9      pq.Q(0, 1) | pq.Beamsplitter(theta=np.pi / 3, phi = np.pi / 4)
10
11     # Measurement
12     pq.Q(0, 1) | pq.ParticleNumberMeasurement()
13
14 # Choosing a simulator
15 simulator = pq.GaussianSimulator(d=2)
16
17 # Execution
18 result = simulator.execute(program, shots=100)
```

Performance benchmark



PiquassoBoost C/C++ extension

To increase computational performance while keeping the benefits coming from the flexibility and the popularity of a high-level Python API the Piquasso simulation package is coming with low-level C++ engines that can be optionally incorporated into the Piquasso framework via lightweight Python C++ extensions interface.

The PiquassoBoost package unifies all the C/C++ components working behind the Piquasso API onto common grounds by utilizing a general development framework responsible for the management of data arrays. Furthermore, it can off-load certain computations to an FPGA-based accelerator backend which operates according to the Data-Flow Engine (DFE) paradigm.

On the left, a performance benchmark is shown comparing the DFE permanent calculator implementation and the fastest CPU implementations available today. We compared the performance of our implementation provided in the Piquasso Boost library to the implementation of TheWalrus version 0.16.2 package also having implemented parallelized C++ engines to evaluate the permanent function. (Newer versions on TheWalrus do not contain C++ engines, nor extended precision implementations.)

Web Application

The Piquasso web application is a dynamic drag-and-drop circuit called the Composer, which can be used to compose quantum circuits. The created quantum program can be sent to the server for simulation. After the simulation, one can acquire information about the particle number detection probabilities, the Wigner function or the canonical momentum and position distribution. One can also export the generated data in a JSON file for further processing and the application even generates a Python code, which could be run using the installed Piquasso/PiquassoBoost libraries.

On the right one can see a basic circuit created by the Composer. Users can use a similar set of gates as in the installed libraries. After running the simulation, one is able to visualize their data e.g. by plotting a Wigner function. Piquasso users are also able to browse, find and discover new circuits in Piquasso's public repository. Using this sharing functionality users can extend the community's knowledge base with their ideas.

Piquasso web interface