The Journal of Open Source Software

Qiskit Experiments: A Python package to characterize and calibrate quantum computers

Naoki Kanazawa [1], **Daniel J. Egger** [2¶], **Yael Ben-Haim** [3], **Helena Zhang** [4], **William E. Shanks** [4], **Gadi Aleksandrowicz** [3], **and Christopher J. Wood** [4]

1 IBM Quantum, IBM Research Tokyo, Tokyo, Japan **2** IBM Quantum, IBM Research Europe - Zurich, Ruschlikon, Switzerland **3** IBM Quantum, IBM Research Israel, Haifa, Israel **4** IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, USA ¶ Corresponding author

DOI: 10.21105/joss.05329

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @nunezco2
- @goerz
- @TejasAvinashShetty

Submitted: 15 March 2023
Published: 19 April 2023

Summary

Qiskit Experiments is a Python package for designing and running quantum computing experiments with a focus on calibration and characterization of quantum devices. It consists of a general purpose experiments framework which can be used by researchers to rapidly implement new experiments and a library of common experiments for calibration, characterization, and verification of quantum devices.

The core framework of `Qiskit Experiments` consists of three parts. (i) An experiment class defines the quantum circuits to run. (ii) A data container class named `ExperimentData` stores the data measured during the execution of the quantum circuits. (iii) An analysis class, attached to each experiment, defines how to analyze the measured data. The analysis also attaches its results, such as fit results and figures, to the data container. Figure 1 summarizes this framework. Importantly, this framework can interface with services to store, load, and share data. The library of experiments includes common quantum computing experiments such as Randomized Benchmarking (Magesan et al., 2011), Quantum State and Process Tomography (Banaszek et al., 1999), Quantum Volume (Cross et al., 2019), and gate error amplifying calibration sequences (Tornow et al., 2022).

`Qiskit Experiments` is based on Qiskit (Anis et al., 2021), a general purpose Python library for programming quantum computers and simulators. It uses many other open source packages. These include Numpy (Harris et al., 2020) for fast numerical computing, lmfit (Newville et al., 2014) to fit complex models to data, CVXPY (Diamond & Boyd, 2016) for convex optimization, Matplotlib (Hunter, 2007) for plotting, and uncertainties (Lebigot, n.d.) to provide measurements with a mean and a standard deviation.

Statement of need

Quantum computing processes information following the laws of quantum mechanics. Quantum computers, like classical computers, must be programmed to perform quantum computations. A quantum computer consists of qubits which store information in quantum states, along with additional hardware elements, such as resonators to couple, control, and readout the qubits. The different elements in the quantum hardware have properties that must be characterized to calibrate the quantum gates that process the information. Furthermore, the quality of these gates has to be benchmarked to measure the overall performance of the quantum computer. This characterization and calibration requires an extensive set of experiments and analysis routines.

Quantum development software packages such as Qiskit, ReCirq (Quantum AI team and collaborators, 2020), tKet (Sivarajah et al., 2020), and Forest (Smith et al., 2016) are part of the quantum stack to execute quantum circuits on hardware. They also enable high-level applications that abstract away the quantum hardware. Forest-benchmarking (Gulshen et al., 2019) and pyGSTi (Nielsen et al., 2022) are tailored towards benchmarking of quantum hardware. Commercial solutions that provide quantum optimal control as a service now also exist (Ball et al., 2021). However, there is still a need for open-source software that enables researchers and hardware maintainers to easily execute characterization and calibration experiments. Recently, software packages have started to emerge to fill this gap (Pasquale et al., 2023). `Qiskit Experiments` is unique in this perspective as it provides open-source low-level characterization experiments that integrate with pulse-level control (Alexander et al., 2020). `Qiskit Experiments` greatly simplifies the execution of complex experiments and is usable with any hardware exposed as a Qiskit backend. Indeed, a library provides many experiments which run multiple quantum circuits and complex fitting. Crucially, each experiment only requires a few code lines to run with `Qiskit Experiments`. In addition, the base framework of `Qiskit Experiments` provides experimentalists a clear interface to create new experiments. They must (i) define how to construct the circuits, (ii) define the experiment options, and optionally (iii) implement the analysis class, if not already present in the library. In addition, `Qiskit Experiments` provides a calibration framework to manage device calibration. Experiments in the `Qiskit Experiments` library and custom extensions built on top of the framework have been used to explore measurements without qubit reset (Tornow et al., 2022), benchmarking (Amico et al., 2023), positive operator value measures (Fischer et al., 2022), quantum states (Hamilton et al., 2022), and time-evolutions (Greenaway et al., 2022), as well as calibrate gates (Vazquez et al., 2022).

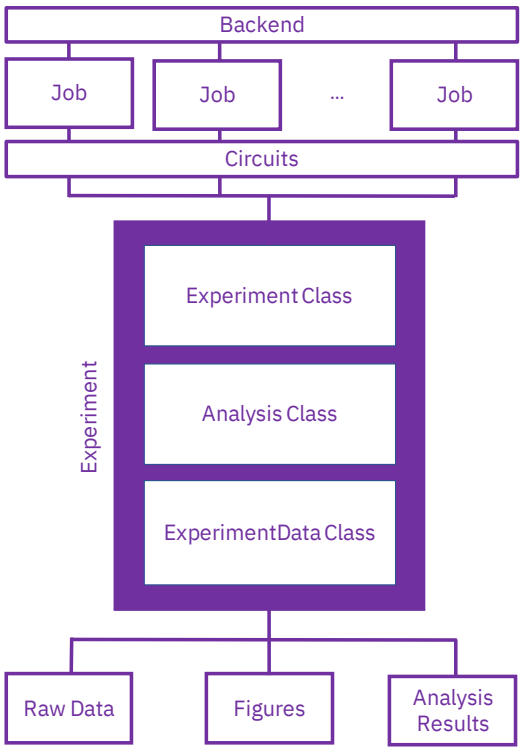

Figure 1: Conceptual framework of Qiskit Experiments. The circuits are run as jobs on the quantum backends. If an experiment exceeds the maximum circuit limit per job it is broken down in multiple jobs. The raw data, figures and analysis results are contained in the `ExperimentData` class.

Example usage

`Qiskit Experiments` can run, for example, a Quantum Volume (QV) measurement (Cross et al., 2019). Evaluating the QV of a quantum computer requires executing random SU(4) circuits to quantify the largest quantum circuit with equal width and depth that can be successfully run. A depth d QV circuit is successful if it has mean heavy-output probability greater than two-thirds with a confidence level exceeding 0.977, and at least 100 trials have been run. The ideal heavy-outputs are determined from a classical simulation of the quantum circuits. `Qiskit Experiments` only requires a few lines of code to run this standardized yet complex experiment, as shown in the online documentation. The analysis classes of existing experiments automatically generate key figures with customizable visualization options, as exemplified by the QV plot in Figure 2.

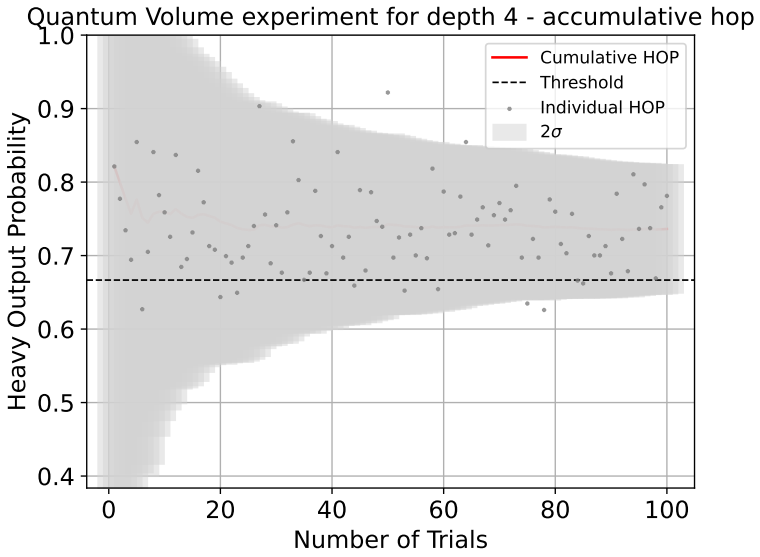

Figure 2: Example result of a quantum volume measurement carried out with `Qiskit Experiments` on a noisy simulator. The dashed line shows the two-thirds threshold. Each dot shows an execution of a randomized quantum circuit aggregated over many shots. The shaded area is a 2σ confidence interval.

Documentation

`Qiskit Experiments` documentation is available at https://qiskit.org/documentation/experiments. The documentation also includes experiment manuals that show how to run experiments such as the Quantum Volume presented above.

Acknowledgements

We acknowledge contributions from the Qiskit Community and feedback from our users.

References

Alexander, T., Kanazawa, N., Egger, D. J., Capelluto, L., Wood, C. J., Javadi-Abhari, A., & McKay, D. C. (2020). Qiskit pulse: Programming quantum computers through the cloud with pulses. *Quantum Sci. Technol.*, *5*(4), 044006. https://doi.org/10.1088/2058-9565/aba404

Amico, M., Zhang, H., Jurcevic, P., Bishop, L. S., Nation, P., Wack, A., & McKay, D. C. (2023). *Defining standard strategies for quantum benchmarks*. https://doi.org/10.48550/ARXIV.2303.02108

Anis, M. S., Mitchell, A., Abraham, H., Offei, A., Agarwal, R., Agliardi, G., Aharoni, M., Akhalwaya, I. Y., Aleksandrowicz, G., & et al. (2021). *Qiskit: An open-source framework for quantum computing*. https://doi.org/10.5281/zenodo.2573505

Ball, H., Biercuk, M. J., Carvalho, A. R. R., C., J., Hush, M., Castro, L. A. D., Li, L., Liebermann, P. J., Slatyer, H. J., Edmunds, C., Frey, V., Hempel, C., & Milne, A. (2021). Software tools for quantum control: Improving quantum computer performance through noise and error suppression. *Quantum Sci. Technol.*, *6*(4), 044011. https://doi.org/10.1088/2058-9565/abdca6

Banaszek, K., D'Ariano, G. M., Paris, M. G. A., & Sacchi, M. F. (1999). Maximum-likelihood estimation of the density matrix. *Phys. Rev. A*, *61*, 010304. https://doi.org/10.1103/PhysRevA.61.010304

Cross, A. W., Bishop, L. S., Sheldon, S., Nation, P. D., & Gambetta, J. M. (2019). Validating quantum computers using randomized model circuits. *Phys. Rev. A*, *100*, 032328. https://doi.org/10.1103/PhysRevA.100.032328

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, *17*(83), 1–5.

Fischer, L. E., Miller, D., Tacchino, F., Barkoutsos, P. Kl., Egger, D. J., & Tavernelli, I. (2022). Ancilla-free implementation of generalized measurements for qubits embedded in a qudit space. *Phys. Rev. Res.*, *4*, 033027. https://doi.org/10.1103/PhysRevResearch.4.033027

Greenaway, S., Smith, A., Mintert, F., & Malz, D. (2022). *Analogue quantum simulation with fixed-frequency transmon qubits*. arXiv. https://doi.org/10.48550/ARXIV.2211.16439

Gulshen, K., Combes, J., Harrigan, M. P., Karalekas, P. J., Silva, M. P. da, Alam, M. S., Brown, A., Caldwell, S., Capelluto, L., & et al. (2019). *Forest Benchmarking: QCVV using PyQuil*. https://doi.org/10.5281/zenodo.3455847

Hamilton, K. E., Laanait, N., Francis, A., Economou, S. E., Barron, G. S., Yeter-Aydeniz, K., Morris, T., Cooley, H., Kang, M., & et al. (2022). *An entanglement-based volumetric benchmark for near-term quantum hardware*. arXiv. https://doi.org/10.48550/ARXIV.2209.00678

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., & et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Comput. Sci. Engineer.*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Lebigot, E. O. (n.d.). *Uncertainties: A python package for calculations with uncertainties*. https://pythonhosted.org/uncertainties/

Magesan, E., Gambetta, J. M., & Emerson, J. (2011). Scalable and robust Randomized Benchmarking of quantum processes. *Phys. Rev. Lett.*, *106*, 180504. https://doi.org/10.1103/PhysRevLett.106.180504

Newville, M., Stensitzki, T., Allen, D. B., & Ingargiola, A. (2014). *LMFIT: Non-linear least-square minimization and curve-fitting for Python*. https://doi.org/10.5281/zenodo.11813

Nielsen, E., Seritan, S., Proctor, T., Rudinger, K., Young, K., Russo, A., Blume-Kohout, R., Kelly, R. P., Gamble, J. K., & Saldyt, L. (2022). *pyGSTio/pyGSTi: Version 0.9.10.1*. https://doi.org/10.5281/zenodo.6363115

Pasquale, A., Efthymiou, S., Ramos-Calderer, S., Wilkens, J., Roth, I., & Carrazza, S. (2023). *Towards an open-source framework to perform quantum calibration and characterization*. https://arxiv.org/abs/2303.10397

Quantum AI team and collaborators. (2020). *ReCirq*. Zenodo. https://doi.org/10.5281/zenodo.4091470

Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., & Duncan, R. (2020). T|ket : A retargetable compiler for NISQ devices. *Quantum Sci. Technol.*, *6*(1), 014003. https://doi.org/10.1088/2058-9565/ab8e92

Smith, R. S., Curtis, M. J., & Zeng, W. J. (2016). A practical quantum instruction set architecture. In *arXiv preprint arXiv:1608.03355*.

Tornow, C., Kanazawa, N., Shanks, W. E., & Egger, D. J. (2022). Minimum quantum run-time characterization and calibration via restless measurements with dynamic repetition rates. *Phys. Rev. Appl.*, *17*, 064061. https://doi.org/10.1103/PhysRevApplied.17.064061

Vazquez, A. C., Egger, D. J., Ochsner, D., & Woerner, S. (2022). *Well-conditioned multi-product formulas for hardware-friendly hamiltonian simulation*. arXiv. https://doi.org/10.48550/ARXIV.2207.11268