TIF-UNIMI-2024-7

Beyond full-state vector simulation with `Qibo`

Andrea Pasquale[1,2], Andrea Papaluca[3], Renato M. S. Farias[1,4], Matteo Robbiati[2,5], Edoardo Pedicillo[1,2], and Stefano Carrazza[1,2,5]

[1]Quantum Research Center, Technology Innovation Institute, Abu Dhabi, UAE
[2]TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano
[3]School of Computing, Australian National University, Canberra, ACT, Australia
[5]European Organization for Nuclear Research (CERN), Geneva 1211, Switzerland
[4]Instituto de Física, Universidade Federal do Rio de Janeiro, P.O. Box 68528, Rio de Janeiro, Rio de Janeiro 21941-972, Brazil

E-mail: andrea.pasquale@unimi.it

Abstract. In this proceedings, we present two new quantum circuit simulation protocols recently added as optional backends to `Qibo`, an open-source framework for quantum simulation, hardware control and calibration. We describe the current status of the framework as for version `0.2.9`. In detail, the two new backends for Clifford and tensor networks simulation are presented and benchmarked against the *state-of-the-art*.

1 Introduction

With the significant achievements reached by quantum technologies [1–3], the interest towards quantum computing is consistently growing. However, the noise affecting these early devices is preventing large-scale applications of quantum algorithms. Waiting for quantum computers to reach a level of reliability that allows for full-scale execution of quantum computing routines without limitations on the problem size, it is necessary to continuously improve the simulation tools we have at our disposal. In particular, the development of classical simulation techniques is crucial to overcome the intrinsic representation limit in full statevector simulation, where the memory required to fully represent a qubit system explodes exponentially as the number of qubits increases.

Several open-source software packages for classical simulation of quantum systems are available. While some of them, including `Qiskit` [4] and `Cirq` [5], `cuQuantum` [6], `tket` [7] and `Yao` [8], provide general tools for quantum simulations, other libraries such as `Pennylane` [9] mostly focus on specific applications like quantum machine learning.

In this context, for the last 4 years we have been developing `Qibo` [10], an open-source software framework for quantum computing. At its early stages, `Qibo` was mainly dedicated to full statevector (including density matrix) simulation [11], and managed to achieve performance competitive with state-of-the-art simulators, thanks, also, to a Just-In-Time compilation approach [12]. The current layout of the `Qibo` framework, as of version `0.2.9`, is shown in Fig. 1. `Qibo` provides a language Application Programming Interface (API) to deploy quantum algorithms using either a circuit-based or a quantum annealing-based approach. Moreover, we provide general-purpose tools that are useful in quantum information theory, including calculation of distances among quantum states and quantum channels. `Qibo`'s modular structure enables the deployment of any component of the language API on different software and hardware platforms, which we refer to as *backends*. We are now going to briefly present all the backends available in version `0.2.9`. The `Qibo` package [13] upon installation is equipped with a backend based on `Numpy` [14], which provides adequate performances for simulating circuits with a relatively low number

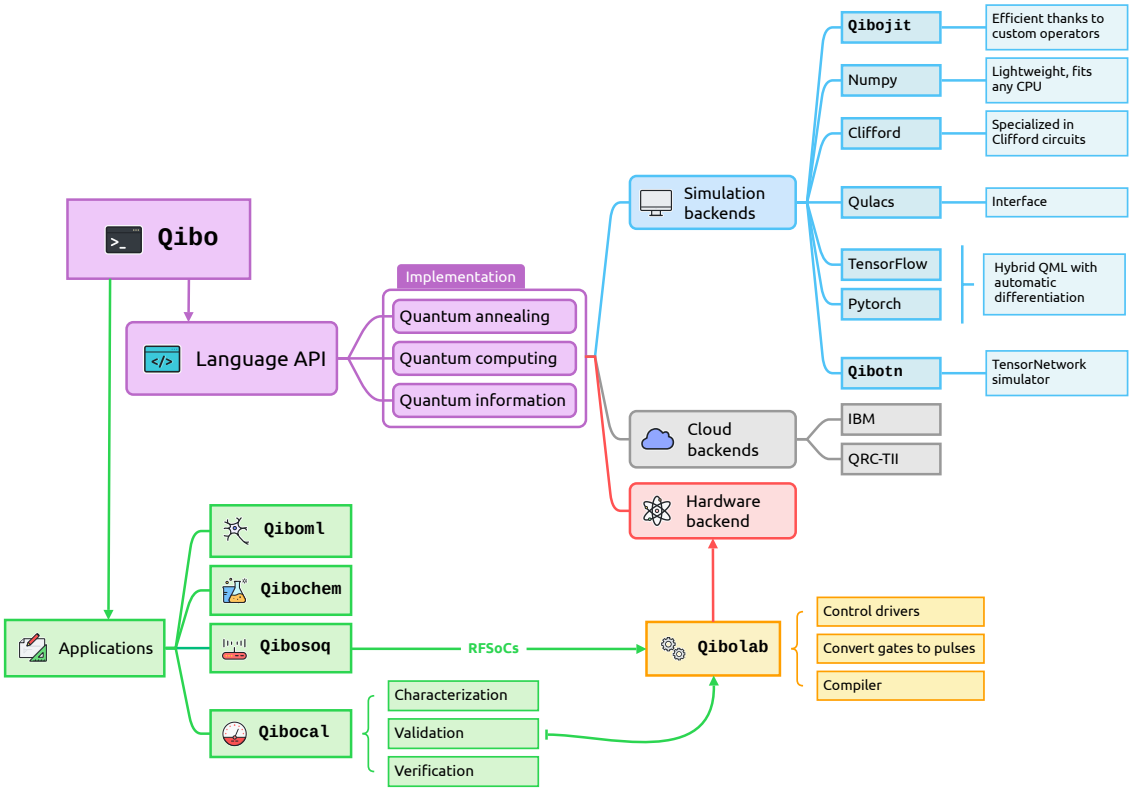


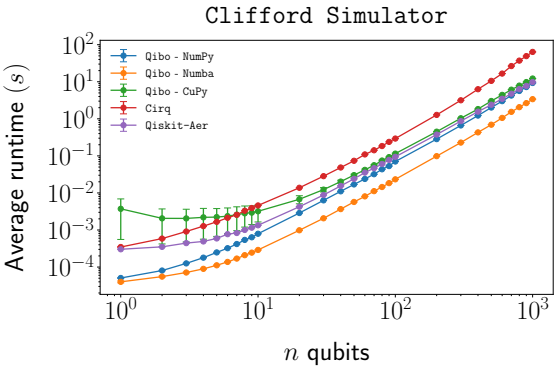
Figure 1: `Qibo` framework in version `0.2.9` [13].

of qubits n (i.e. $n \leq 20$). We also provide a more efficient general purpose simulator, called `Qibojit`, which supports hardware acceleration. More specifically, `Qibojit` allows multi-threaded CPU execution using `Numba` [15], while GPU support is enabled both through `Cupy` [16] and through compatibility with NVIDIA's `CuQuantum` [6] library. To improve the performance, `Qibojit` defines custom operators that exploit the sparsity of the matrix representation of several quantum gates that are often used in quantum computing. Such improvements are also reflected in the quantum annealing approach when the Trotter decomposition is used. We also provide backends supporting automatic differentiation specifically designed for quantum machine learning applications, including one based on `TensorFlow` [17] primitives, and a recently added one based on the popular machine learning framework `Pytorch` [18]. The possibility of integrating automatic differentiation tools within such a modular environment can be exploited to develop and test both pure quantum and hybrid classical-quantum machine learning algorithms [19–25].

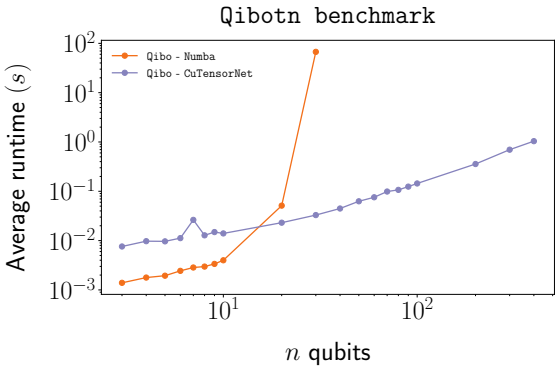
As a middleware software package, `Qibo` also provides a backend for quantum hardware execution. This backend is called `Qibolab`, which defines a dedicated API to perform instrument control, driver operations, as well as compilation of `Qibo` circuits into customizable native gates sets. Alongside `Qibolab`, we provide a dedicated package to characterize and calibrate self-hosted quantum devices: `Qibocal` [26]. Besides simulation and hardware backends, there is an ongoing effort to provide cloud access, giving users the possibility to manage their self-hosted quantum hardware by using `Qibolab`'s hardware control tools, or to deploy quantum circuits on well-known quantum cloud providers, including IBM Quantum [4] and IonQ [27].

This was a brief overview to showcase the wide modularity and diversity that `Qibo` provides. As a demonstration of the ongoing effort to continuously support new platforms and extend `Qibo`'s compatibility with *state-of-the-art* quantum computing software stacks we recently added a backend that interfaces `Qibo` with `Qulacs` [28].

Below, we focus on two simulation backends recently added to the `Qibo` framework, which are dedicated to tensor network simulation of quantum circuits and fast simulation of Clifford circuits.



(a) Simulation of Clifford circuits with an increasing number of qubits. For each point we take the average over 100 different randomly generated circuits. Circuits were generated following Ref. [38], which guarantees an uniform distribution of the generated n-qubit Clifford operators. We did not include measurements in this benchmark.



(b) Total simulation time between `Qibojit` and `Qibotn` for a variational circuit. See [12] for the specific circuit employed. The execution is performed on a `NVIDIA` A100 GPU using NVIDIA and on AMD EPYC 7713.

Figure 2: Benchmarks showing the performances of the Clifford simulator and `Qibotn`.

2 Simulation of Clifford circuits

With full statevector simulation, it is only possible to simulate circuits up to a limited number of qubits: John Preskill describes this limitation in Ref. [29] introducing the approximate yet explanatory term of *50-qubit barrier*. The main limitations are imposed by huge memory requirements and very long computation times [30]. Over the last decades, researchers have studied extensively *Clifford circuits*, a class of circuits that can be simulated in polynomial times [31]. The quantum states generated by Clifford circuits are called *stabilizers states* [32], and have many applications in quantum information theory [33–37]. As current quantum hardware gets closer to the requirements needed for quantum error correction[3], such circuits have started to receive more attention, leading to the development of techniques to classically simulate them.

Based on this, a new backend has been developed in `Qibo` which focuses on fast and efficient simulation of Clifford circuits. The implementation in `Qibo` is based on the phase-space formalism introduced in Ref. [31]. From the perspective of code design, the implementation makes full use of `Qibo`'s modularity, allowing for alternative backends to be easily plugged in. The basic implementation proposed is based on `Numpy` primitives. This enables the single-threaded simulation of Clifford circuits on CPUs. By taking advantage of `Qibojit`, we also provide implementations based on `Numba` and `Cupy`. The integration with the `Numba` backend for multi-threaded computations on CPUs is done via custom kernels that are compiled *just-in-time*. For GPU integration, we make use of custom CUDA C kernels through `Cupy`.

To evaluate our implementation, we compare our Clifford simulator with Clifford simulators available in `Qiskit` and `Cirq`. The results are shown in Fig. 2a. We tested all the simulators on the same dataset consisting of 100 random Clifford circuits for each system syze n. These circuits were sampled uniformly following Ref. [38]. We ran the CPU benchmarks on an AMD EPYC 7773X processor and the GPU benchmarks on a NVIDIA RTX Quadro a6000. Our `Numpy` and `Cupy` backends asymptotically approach performance similar to `Qiskit`, whereas our `Numba`-based implementation displays an advantage over the whole range of qubits considered. For instance, for $n = 1000$, our `Numba` backend is up to one order of magnitude faster than `Qiskit`, and almost two order of magnitudes faster than `Cirq`.

Here, we make a remark about our GPU implementation. As expected, the overhead of copying the data from the host to the device is dominating the results for a small number of qubits. However, an improvement due to the huge parallelization capabilities of GPUs is expected to appear at $n \gtrsim 2^{11}$. Since we benchmarked average runtimes up to 1000 qubits, this "crossing" in performance was not observed yet, making further investigation of bigger systems a necessity.

3 TensorNetwork simulation using `Qibo`

After showing that for specific type of quantum circuits we can reduce the computational time, we now introduce a second popular approach to simulate large quantum circuits: classical approximation

methods. A popular method for approximating quantum circuits are tensor networks (TN) [39], which represents states or operators as network of smaller tensors reducing both memory and computational requirements. They are successfully used, for instance, to solve one-dimensional strongly-correlated quantum systems [40]. On the other hand, the effectiveness of representing large-scale systems comes at the cost of introducing truncation errors, which make these techniques less effective when there is the need to know the quantum state more exhaustively.

Within the `Qibo` framework, we have recently developed `Qibotn`, a `Qibo` subpackage which enables to execute quantum circuits using tensor network like computations, allowing to support large-scale simulation of quantum circuits in `Qibo`. `Qibotn` interfaces `Qibo` with state-of-the-art quantum TN simulation libraries such as `CuTensorNet` [6] from NVIDIA and `quimb` [41]. Both Matrix Product States and generic TN are supported. `Qibotn` is designed to support High Performance Computing configurations including single node GPUs, as well as multi-node multi-GPU configuration using Message Passing Interface or the NVIDIA Collective Communication Library (NCCL) from `NVIDIA`.

To showcase the capabilities of the library we perform comparison between `Qibojit` and `Qibotn` performances in Fig.2b. We execute a variational quantum circuit on a `NVIDIA` A100 GPU for several number of qubits. As expected, using full statevector simulation with `Qibojit` it possible to run only up to 40, while using `Qibotn` we show how the curve flattens and we observe that we are able to simulate a variational circuit with up to 400 qubits. Moreover, given the slow rise of the curve we expect to increase the number of qubits with appropriate memory requirements.

4 Outlook

In this proceedings, we have described the latest updates available in `Qibo 0.2.9`. After a brief overview on all modules currently available in the `Qibo` framework, we have put the focus on two new simulation methodologies: Clifford simulation and Tensor Networks (TN). We have shown that our Clifford simulator is competitive with *state-of-the-art* libraries. We further demonstrated that, despite its early development stage, our TN implementation is able to simulate efficiently circuits with up to 400 qubits. Future developments of the `Qibo` framework include having a dedicated module to perform QML algorithms, which we refer to as `Qiboml`, and we are looking forward to expand `Qibo` to support also quantum chemistry, as well as Quantum optimization problems. Finally, although `Qibo` has been developed in `Python`, we are looking to separate `Qibo` core elements to take advantage of the better performance offered by other languages, e.g. `C++` and `Rust`.

5 Acknowledgements

This project is supported by `TII`'s Quantum Research Center. The authors thank all `Qibo` contributors for helpful discussion and Liwei Yang and Andy Tan Kai Yong for their support in developing `Qibotn`. M.R. is supported by `CERN`'s Quantum Technology Initiative (`QTI`) through the Doctoral Student Program.

References

[1] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Wei, Ewout Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, and Abhinav Kandala. Evidence for the utility of quantum computing before fault tolerance. *Nature*, 618:500–505, 06 2023. doi: 10.1038/s41586-023-06096-3.

[2] Frank Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019. doi: 10.1038/s41586-019-1666-5. URL `https://doi.org/10.1038/s41586-019-1666-5`.

[3] Sebastian Krinner et al. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, May 2022. ISSN 1476-4687. doi: 10.1038/s41586-022-04566-8. URL `http://dx.doi.org/10.1038/s41586-022-04566-8`.

[4] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit, 2024.

[5] Cirq Developers. Cirq, May 2024. URL `https://doi.org/10.5281/zenodo.11398048`.

[6] The cuQuantum development team. Nvidia cuquantum sdk, 2023. URL `https://github.com/nvidia/cuquantum`. BSD-3-Clause License, `https://github.com/NVIDIA/cuQuantum/blob/main/LICENSE`.

[7] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t—ket⟩: a retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003, November 2020. ISSN 2058-9565. doi: 10.1088/2058-9565/ab8e92. URL http://dx.doi.org/10.1088/2058-9565/ab8e92.

[8] Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, and Lei Wang. Yao. jl: Extensible, efficient framework for quantum algorithm design. *Quantum*, 4:341, 2020.

[9] Ville Bergholm et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations, 2022. URL https://arxiv.org/abs/1811.04968.

[10] Stavros Efthymiou, Sergi Ramos-Calderer, Carlos Bravo-Prieto, Adrián Pérez-Salinas, Diego García-Martín, Artur Garcia-Saez, José Ignacio Latorre, and Stefano Carrazza. *Qibo*: a framework for quantum simulation with hardware acceleration. *Quantum Science and Technology*, 7(1):015018, December 2021. ISSN 2058-9565. doi: 10.1088/2058-9565/ac39f5. URL http://dx.doi.org/10.1088/2058-9565/ac39f5.

[11] S. Carrazza, S. Efthymiou, M. Lazzarin, and A. Pasquale. An open-source modular framework for quantum computing. *Journal of Physics: Conference Series*, 2438(1): 012148, February 2023. ISSN 1742-6596. doi: 10.1088/1742-6596/2438/1/012148. URL http://dx.doi.org/10.1088/1742-6596/2438/1/012148.

[12] Stavros Efthymiou, Marco Lazzarin, Andrea Pasquale, and Stefano Carrazza. Quantum simulation with just-in-time compilation. *Quantum*, 6:814, September 2022. ISSN 2521-327X. doi: 10.22331/q-2022-09-22-814. URL http://dx.doi.org/10.22331/q-2022-09-22-814.

[13] Stavros Efthymiou, Renato M. S. Farias, Stefano Carrazza, Andrea Papaluca, Matteo Robbiati, Edoardo Pedicillo, Andrea Pasquale, Simone Bordoni, Alejandro Sopena, Sam-XiaoyueLi, shangtai, Carlos Bravo-Prieto, Alessandro Candido, AdrianPerezSalinas, Yelyzaveta Vodovozova, Sergi Ramos-Calderer, Wen Jun, Diego García-Martín, Marco Lazzarin, Jun Yong Khoo, Jorge J. Martínez de Lejarza, Andrew Wright, Jian Feng Kong, Nicole Zattarin, Ema Puljak, Luca Zilli, Paul, Marek Gluza, and rahul. qiboteam/qibo: Qibo 0.2.9, June 2024. URL https://doi.org/10.5281/zenodo.12577885.

[14] Charles R. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[15] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[16] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. *CuPy*: A numpy-compatible library for nvidia gpu calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. URL http://learningsys.org/nips17/assets/papers/paper_16.pdf.

[17] Martín Abadi et al. *TensorFlow*: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[18] Adam Paszke et al. *PyTorch*: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

[19] Carlos Bravo-Prieto, Julien Baglio, Marco Cè, Anthony Francis, Dorota M. Grabowska, and Stefano Carrazza. Style-based quantum generative adversarial networks for monte carlo events. *Quantum*, 6:777, August 2022. ISSN 2521-327X. doi: 10.22331/q-2022-08-17-777. URL http://dx.doi.org/10.22331/q-2022-08-17-777.

[20] Adrián Pérez-Salinas, Juan Cruz-Martinez, Abdulla A. Alhajri, and Stefano Carrazza. Determining the proton content with a quantum computer. *Physical Review D*, 103 (3), February 2021. ISSN 2470-0029. doi: 10.1103/physrevd.103.034027. URL http://dx.doi.org/10.1103/PhysRevD.103.034027.

[21] Matteo Robbiati, Juan M. Cruz-Martinez, and Stefano Carrazza. Determining probability density functions with adiabatic quantum computing, 2023. URL https://arxiv.org/abs/2303.11346.

[22] Matteo Robbiati, Stavros Efthymiou, Andrea Pasquale, and Stefano Carrazza. A quantum analytical adam descent through parameter shift rule using qibo, 2022. URL https://arxiv.org/abs/2210.10787.

[23] Matteo Robbiati, Alejandro Sopena, Andrea Papaluca, and Stefano Carrazza. Real-time error mitigation for variational optimization on quantum hardware, 2023. URL https://arxiv.org/abs/2311.05680.

[24] Juan M Cruz-Martinez, Matteo Robbiati, and Stefano Carrazza. Multi-variable integration with a variational quantum circuit. *Quantum Science and Technology*, 9(3):035053, June 2024. ISSN 2058-9565. doi: 10.1088/2058-9565/ad5866. URL http://dx.doi.org/10.1088/2058-9565/ad5866.

[25] Simone Bordoni, Denis Stanev, Tommaso Santantonio, and Stefano Giagu. Long-lived particles anomaly detection with parametrized quantum circuits. *Particles*, 6(1):297–311, 2023. ISSN 2571-712X. doi: 10.3390/particles6010016. URL https://www.mdpi.com/2571-712X/6/1/16.

[26] Andrea Pasquale, Stavros Efthymiou, Sergi Ramos-Calderer, Jadwiga Wilkens, Ingo Roth, and Stefano Carrazza. Towards an open-source framework to perform quantum calibration and characterization, 2024. URL https://arxiv.org/abs/2303.10397.

[27] IonQ Inc. IonQ Quantum Cloud, 2024. URL https://ionq.com/quantum-cloud.

[28] Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, Yuria Hiraga, Masahiro Nakadai, Jiabao Chen, Ken M. Nakanishi, Kosuke Mitarai, Ryosuke Imai, Shiro Tamiya, Takahiro Yamamoto, Tennin Yan, Toru Kawakubo, Yuya O. Nakagawa, Yohei Ibe, Youyuan Zhang, Hirotsugu Yamashita, Hikaru Yoshimura, Akihiro Hayashi, and Keisuke Fujii. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum*, 5:559, October 2021. ISSN 2521-327X. doi: 10.22331/q-2021-10-06-559. URL http://dx.doi.org/10.22331/q-2021-10-06-559.

[29] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, August 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL http://dx.doi.org/10.22331/q-2018-08-06-79.

[30] George F. Viamontes, Igor L. Markov, and John P. Hayes. Improving gate-level simulation of quantum circuits, 2003. URL https://arxiv.org/abs/quant-ph/0309060.

[31] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5), November 2004. ISSN 1094-1622. doi: 10.1103/physreva.70.052328. URL http://dx.doi.org/10.1103/PhysRevA.70.052328.

[32] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction and orthogonal geometry. *Physical Review Letters*, 78(3):405–408, January 1997. ISSN 1079-7114. doi: 10.1103/physrevlett.78.405. URL http://dx.doi.org/10.1103/PhysRevLett.78.405.

[33] Daniel Gottesman. Stabilizer codes and quantum error correction, 1997. URL https://arxiv.org/abs/quant-ph/9705052.

[34] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Physical Review A*, 57(1):127–137, January 1998. ISSN 1094-1622. doi: 10.1103/physreva.57.127. URL http://dx.doi.org/10.1103/PhysRevA.57.127.

[35] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, 71(2), feb 2005. ISSN 1094-1622. doi: 10.1103/physreva.71.022316. URL http://dx.doi.org/10.1103/PhysRevA.71.022316.

[36] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1), jan 2008. ISSN 1094-1622. doi: 10.1103/physreva.77.012307. URL http://dx.doi.org/10.1103/PhysRevA.77.012307.

[37] Hsin-Yuan Huang, Richard Kueng, and John Preskill. Predicting many properties of a quantum system from very few measurements. *Nature Physics*, 16(10):1050–1057, jun 2020. ISSN 1745-2481. doi: 10.1038/s41567-020-0932-7. URL http://dx.doi.org/10.1038/s41567-020-0932-7.

[38] Sergey Bravyi and Dmitri Maslov. Hadamard-free circuits expose the structure of the clifford group. *IEEE Transactions on Information Theory*, 67(7):4546–4563, July 2021. ISSN 1557-9654. doi: 10.1109/tit.2021.3081415. URL http://dx.doi.org/10.1109/TIT.2021.3081415.

[39] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell, 2017. URL https://arxiv.org/abs/1708.00006.

[40] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, January 2011. ISSN 0003-4916. doi: 10.1016/j.aop.2010.09.012. URL http://dx.doi.org/10.1016/j.aop.2010.09.012.

[41] Johnnie Gray. *quimb*: a python library for quantum information and many-body calculations. *Journal of Open Source Software*, 3(29):819, 2018. doi: 10.21105/joss.00819.