A Software Simulator for Noisy Quantum Circuits

Himanshu Chaudhary[1], Biplab Mahato[2], Lakshya Priyadarshi[3], Naman Roshan[4], Utkarsh[5], and Apoorva D. Patel[6]

[1] *Indian Institute of Science, Bangalore 560012*
[2] *Indian Institute of Science, Bangalore 560012*
[3] *Institute of Engineering and Technology, Lucknow 226021*
[4] *Indian Institute of Technology, Kanpur 208016*
[5] *Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology Hyderabad, Hyderabad 500032, Telangana, India.*
[6] *Centre for High Energy Physics, Indian Institute of Science, Bangalore 560012*

Abstract. We have developed a software library that simulates noisy quantum logic circuits. We represent quantum states by their density matrices, and incorporate possible errors in initialisation, logic gates, memory and measurement using simple models. Our quantum simulator is implemented as a new backend on IBM's open-source Qiskit platform.

Keywords: Software Simulator, Quantum Simulations, Density Matrices, NISQ

Arxiv E-print: arXiv:1908.05154 **[quant-ph]**

Codebase: https://github.com/indian-institute-of-science-qc/qiskit-aakash

Software quantum simulators are the programs running on classical parallel computer platforms, and can model and benchmark NISQ systems. It is not possible to classically simulate larger qubit systems due to exponential growth in the Hilbert space size. They are portable, and can be easily distributed over existing computational facilities world-wide. They are therefore an excellent way to attract students to the field of quantum technology, providing a platform to acquire the skills of 'programming' as well as 'designing' quantum processors. With this aim, we have constructed a software library for simulating noisy quantum logic circuits. Essentially, our quantum simulator is an open-source software written in Python, which is added as a new backend to IBM's Qiskit platform. That extends the existing Qiskit capability, while retaining the convenience (e.g. portability, documentation, graphical interface) of the Qiskit format.

1. IMPLEMENTATION

The most general description of a quantum system is in terms of its density matrix ρ, which evolves according to a linear completely positive trace preserving map known as the superoperator, provides an ensemble description of the quantum system, and so is inherently probabilistic, in contrast to the state vector description that can describe individual experimental system evolution. Our simulator is based on describing and evolving a system in the density matrix formulation. We express the density matrix of an n-qubit quantum register in the orthogonal Pauli basis:

$$\rho = \sum_{i_1,i_2,\ldots,i_n} a_{i_1 i_2 \ldots i_n} (\sigma_{i_1} \otimes \sigma_{i_2} \otimes \ldots \otimes \sigma_{i_n})$$

Quantum dynamics is linear in this picture, and hence, any operation - quantum gates, errors and projective measurements are efficiently implemented in the software using linear algebra vector instructions.

2. CIRCUIT OPTIMISATION

To mitigate the deterioration of open quantum systems in time we try to reduce the total execution time of a quantum program as much as possible. To do this we restructure the quantum circuit using a merging-partitioning algorithm, s.t. it merges consecutive single-qubit rotations that we find in to a single one (e.g. u3 ∗ u3 → u3), using SU(2) group composition rules, and then arrange the complete list of final instructions in to a set of partitions, such that all operations in a single partition can be executed during a single clock step. In particular, this procedure puts logic gate operations and measurement operations in separate partitions.

3. THE NOISY EVOLUTION

The main achievement of our simulator is the ability to simulate noisy quantum systems, using simple error models. We assume that the environment has an independent effect on each qubit to include the environmental noise in the simulator at various stages of the program execution: initialisation error (due to thermalization), logic gate execution error, measurement error (due to depolarization), and memory errors (due to decoherence, amplitude decay and thermalization).

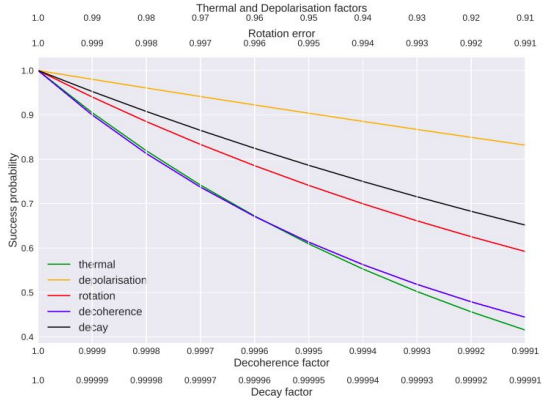

Figure 1 Success probability of the binary addition program, 110+11=1001, as a function of different types of errors. For easy comparison, multiple parameters are plotted along the X-axis with different scales: thermal factor p (green), depolarisation factor d1 (orange), rotation error parameter r (red), decoherence factor f (blue) and decay factor g (black). In all cases, the success probability is observed to decrease exponentially.

The final results are probability distributions over the possible outcomes of the algorithm, and their stability against variations of the error parameters can be explicitly checked. The distributions can be easily visualised using various types of plots, and we expect exponential deterioration of the quantum signal with increasing errors.

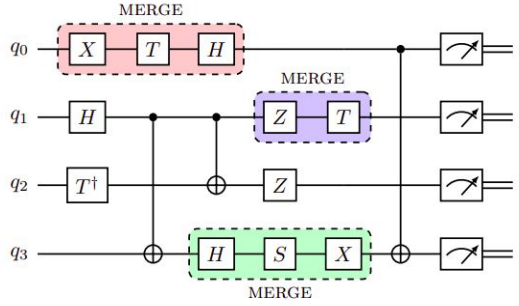

Figure 2 Merging operation of single qubit gates for a quantum circuit. This is the first step our merging-partitioning algorithm

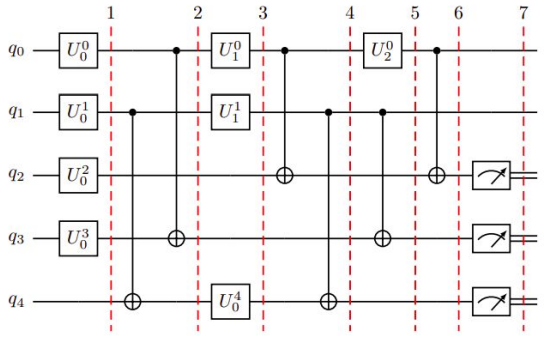

Figure 3 The partitioned logic circuit for a quantum circuit. This is the second step our merging-partitioning algorithm.

4. REFERENCES

[1] J. Preskill, Lecture Notes for the Course on Quantum Computation, http://www.theory.caltech.edu/people/preskill/ph219/

[2] M.A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information, (Cambridge University Press, 2000).

[3] J. Preskill, Quantum Computing in the NISQ Era and Beyond, Quantum 2 (2018) 79.

[4] P.J. Coles et al., Quantum Algorithm Implementations for Beginners, arXiv:1804.03719.

[5] R. LaRose, Overview and Comparison of Gate Level Quantum Software Platforms, arXiv:1807.02500.

[6] Qiskit: An Open-source Framework for Quantum Computing, 10.5281/zenodo.2562110 (2019).