Is Measurement Enough? Rethinking Output Validation in Quantum Program Testing

Jiaming Ye[1], Xiongfei Wu[2], Shangzhou Xia[3], Fuyuan Zhang[4], Jianjun Zhao[3]
[1]Southwest Jiaotong University, China [2]University of Luxembourg, Luxembourg [3]Kyushu University, Japan
[4]Zhejiang University, China

***Abstract*—As quantum computing continues to emerge, ensuring the quality of quantum programs has become increasingly critical. Quantum program testing has emerged as a prominent research area within the scope of quantum software engineering. While numerous approaches have been proposed to address quantum program quality assurance, our analysis reveals that most existing methods rely on measurement-based validation in practice. However, due to the inherently probabilistic nature of quantum programs, measurement-based validation methods face significant limitations.**

To investigate these limitations, we conducted an empirical study of recent research on quantum program testing, analyzing measurement-based validation methods in the literature. Our analysis categorizes existing measurement-based validation methods into two groups: distribution-level validation and output-value-level validation. We then compare measurement-based validation with statevector-based validation methods to evaluate their pros and cons. Our findings demonstrate that measurement-based validation is suitable for straightforward assessments, such as verifying the existence of specific output values, while statevector-based validation proves more effective for complicated tasks such as assessing the program behaviors.

I. Introduction

The growing applications of quantum software engineering have increased the need for quality assurance of quantum programs, leading researchers to propose various testing approaches, such as search-based testing [1], [2], combinatorial testing [3], [4], and metamorphic testing [5], [6]. In addition, various types of assertion have been proposed, including dynamic assertions [7], [8] and statistical assertions [9], opening new opportunities and challenges to advance the quality assurance of quantum programs.

Our investigation of state-of-the-art quantum program testing approaches reveals a critical commonality: most validation methods rely on analyzing the outputs obtained from the measurement results, namely *measurement-based validation*. In contrast, statevector-based validation methods are rarely adopted in existing approaches. The working processes of these two validation methods are illustrated in Figure 1. Unlike measurement-based validation, statevector-based validation does not require adding measurement operators to the circuit and directly extracts results without repeatedly sampling measurements.

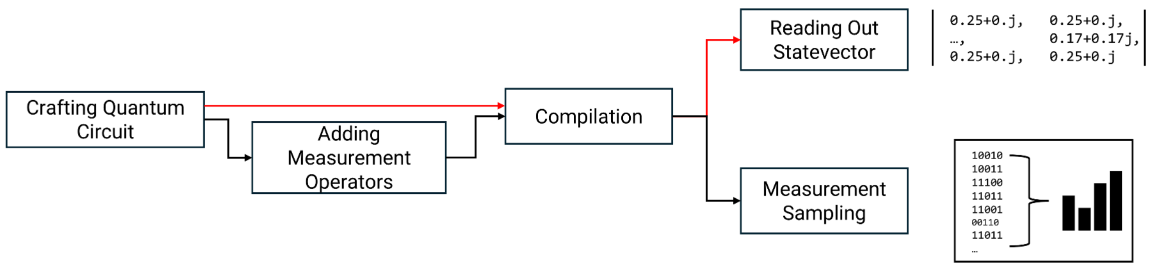
Fuyuan Zhang is with the State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China. This work was supported in part by JSPS KAKENHI Grant No. JP24K14908, and in part by JST SPRING Grant No. JPMJSP2136.

Through our experiments and discussions with Qiskit developers, we identified a severe limitation of measurement-based validation in testing practice. Specifically, due to the probabilistic nature of quantum computing, the appearance of each output value in a single execution follows a particular probability distribution. Existing testing approaches extract measurement values and apply statistical analysis techniques, such as the Kolmogorov-Smirnov test [10], [11] and cross entropy [12], [13], for output validation. However, determining the sufficient number of measurement shots to accurately capture the underlying probability distribution remains inherently challenging. Consequently, existing approaches often use "undersampled" distribution results for validation, particularly in differential testing, where distribution results are used to compare program behaviors. This raises fundamental questions: *Are existing measurement-based validation methods sufficient for quantum program testing? Can statevector-based validation complement the shortcomings of existing approaches?*

To address these questions, we conduct a comprehensive survey of recent quantum program testing approaches to investigate how measurement-based validation participates in testing approaches. Specifically, we categorize existing testing approaches into distribution-level and output-value-level validation methods to answer our first research question (**RQ1**): *In what ways do current testing approaches utilize measurement outcomes to assess program correctness?* Subsequently, we perform an in-depth comparison between statevector-based and measurement-based validation to reveal appropriate usage scenarios for both methods and discuss practical applications of statevector-based approaches, addressing our second research question (**RQ2**): *What are the pros and cons of each validation method? How should these validation methods be appropriately applied in practice?* Based on our findings, we recommend using statevectors for complex validation tasks and measurement-based validation for simple validation requirements.

Contributions. Our main contributions are as follows:

1) We conduct a comprehensive analysis of the validation methods employed in state-of-the-art quantum program testing approaches.
2) We propose using statevectors as a complementary approach to measurement-based validation in quantum program testing.
3) We provide a detailed comparison of both validation methods and provide practical guidelines for their appropriate application.



```
0.25+0.j,   0.25+0.j,
…,          0.17+0.17j,
0.25+0.j,   0.25+0.j
```

```
10010
10011
11100
11011
11001
00110
11011
…
```

Fig. 1: The comparison of the process of reading out the statevector (in red arrow) and measurement sampling (in black arrow).

II. Background

In this section, we introduce the background and concepts related to this work.

A. Qubit, Gates, Circuit, and Quantum Program

A quantum bit (or *qubit*) is the most basic unit in quantum computation. In quantum programs, the operations are represented by quantum logic gates. The gates provide various functions, including rotating a qubit into an arbitrary angle, assigning superposition to qubits, creating controlled connections between qubits, etc. The quantum circuit is the most basic unit for grouping and managing qubits and quantum gates. A quantum circuit is a model for quantum computation [14], in which the qubits, gates, and measurements are organized in a sequence. The quantum program combines the above quantum components and can further include the classical components. The development of modern quantum libraries enables users to manipulate quantum circuits using Python, thus facilitating the integration of the quantum and classical components in quantum programs.

B. Statevector

The statevector in a quantum program represents the complete quantum state of a quantum system as a complex-valued vector in the Hilbert space. For an n-qubit quantum system the statevector $|\psi\rangle$ is defined as:

$$|\psi\rangle = \sum_{|x\rangle \in B} A_x |x\rangle$$

where A_x is the complex probability amplitudes, and the normalization condition $\sum_{|x\rangle \in B} |A_x|^2 = 1$ must be satisfied.

In quantum programs, the statevector is computationally represented as an array of 2^n complex numbers, where each array index corresponds to a specific basis state x. The amplitude at each index determines the probability of measuring the system in that particular computational basis state, with the measurement probability given by $|A_x|^2$.

The statevector of a quantum program is under the action of quantum gate operations as unitary transformations, which can be represented as $|\psi\rangle = U |\psi'\rangle$, where U is a unitary operator preserving the normalization property.

From a quantum program developer's perspective, the statevector serves as the fundamental data structure for quantum state representation in simulators and quantum virtual machines. The statevector enables precise tracking of quantum superposition and entanglement throughout program execution.

C. Measuring the Output of a Quantum Program

To read out the value of a qubit, a measurement operation can be added to the qubit at the end of the program. The measurement operation causes the quantum state to collapse. As explained previously, a qubit state $|\psi\rangle$ exists in a superposition of two orthogonal basis states $|0\rangle$ and $|1\rangle$. After measurement, the state collapses to a definite value of either 0 or 1, with probabilities determined by the amplitude of the superposition.

In the general case of an n-qubit program, we typically add n measurement operations, one for each qubit, at the end of the program. After measurement, the results are read out as an n-bit binary number. Due to the probabilistic nature of the output values, a single measurement is usually insufficient to draw meaningful conclusions. To address this issue, a solution is to repeat the measurement process, which is commonly referred to as repeated m measurements (or samplings). According to the law of large numbers, with an increasing number of samplings, the frequency of observed values is likely to converge to the actual probability distribution. For a quantum program, if the number of samplings is sufficiently large, the distribution of output values should be close to the actual distribution. However, regardless of how well a sampling distribution approximates the actual distribution, it inherently introduces bias and overlooks critical information, thereby causing misjudgments in quantum program testing.

III. Survey

To understand current research trends on validation methods in quantum program testing, we conduct a systematic survey of the recent literature on quantum program testing, as illustrated in Table I. This table presents our comprehensive analysis of the quantum program testing literature, categorizing validation methods using three key dimensions: measurement-based testing, distribution-level validation, and output-value-level validation. Note that distribution-level and output-value-level validation represent two subcategories within measurement-based testing approaches. We employ different circle symbols with varying fill levels to indicate the degree of support each work provides across these dimensions. Specifically, ● means that the validation method is adopted, and ○ means not adopted. Our survey includes mainstream quantum program testing

TABLE I: A survey of quantum program testing literature. Distribution-level approaches treat measurement results as statistical distributions for analysis, whereas output-value-level approaches regard measurement results as plain output values.

Literature	Category	Measurement -based	Distribution -level	Output- value-level
[1]	Search-based	●	●	●
[2]	Search-based	●	●	●
[3]	Combinatorial	●	●	●
[4]	Combinatorial	●	●	●
[5]	Metamorphic Testing	●	●	●
[6]	Metamorphic Testing	●	●	○
[15]	Differential Testing	●	●	○
[16]	Differential Testing	●	●	○
[17]	Fuzzing	●	○	●
[18]	Property-based Testing	●	○	●
[19]	Concolic Testing	●	●	○
[7]	Dynamic Assertion	●	○	●
[8]	Dynamic Assertion	●	○	●
[9]	Statistical Assertion	●	○	●
[20]	Equivalence Check	○	○	○
[21]	Integration Testing	○	○	○

approaches, covering 11 distinct categories including search-based testing, combinatorial testing, metamorphic testing, differential testing, fuzzing, property-based testing, concolic testing, dynamic assertion, statistical assertion, equivalence checking, and integration testing.

Our analysis reveals that most existing approaches adopt measurement-based methods, with only two exceptions: [20] and [21]. These approaches utilize unitary operators or statevectors for validation, thus avoiding the use of measurement-based techniques. For the remaining approaches, they rely on measurement operations to obtain results for subsequent analysis. Based on how measurement results are utilized, we categorize these approaches into two groups: distribution-level and output-value-level validation. Distribution-level validation represents approaches that conduct validation based on the statistical distribution of repeated program executions, while output-value-level validation represents approaches that analyze individual measurement outcomes directly.

Distribution-level validation is used by 8 of the 15 approaches surveyed. Among these, five approaches support both distribution-level and output-value-level analysis, while the remaining three focus exclusively on distribution-level analysis. Our investigation reveals that search-based approaches [1], [2], combinatorial approaches [3], [4], and a metamorphic approach [5] are designed to validate the existence of specific output values. In contrast, other approaches, such as the second metamorphic method [6], differential testing [15], [16], and concolic testing [19] target more complex validation tasks, including program behavior analysis and verification of output equivalence.

Output-value-level validation is adopted by 10 of the 15 approaches. Among these, five approaches support both distribution-level and output-value-level analysis, while the remaining 5 focus solely on output-value-level validation. Specifically, the fuzzing approach [17] uses measurements to assess the probability of occurrence for each output value.

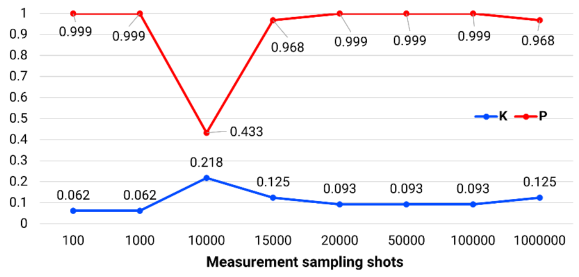


Fig. 2: The K and P values with the increasing measurement sampling shots.

The property-based approach [18] counts the occurrences of the output value for validation purposes. Dynamic assertion approaches [7], [8] and statistical assertion methods [9] focus on detecting the appearance of unexpected output values.

Answer to RQ1: Most state-of-the-art testing approaches for quantum programs rely on measurement-based validation methods. Among these methods, distribution-level validation is used for complex validation tasks such as comparing behaviors between two programs, while output-value-level validation is adopted for simpler tasks, such as verifying the existence of specific output values.

IV. Statevector vs. Measurement

A. *Limitations of Measurement-Based Validation*

Measurement of a quantum program provides a direct method for obtaining program outputs for validation and behavioral assessment. Measurement is typically implemented by adding measurement gates to each qubit at the end of the quantum circuit. Since measurement gates always yield definite values from quantum circuits, the measurement process causes quantum state collapse, resulting in the loss of phase and amplitude information. After sampling (i.e., executing the quantum circuit), an output value is obtained. When multiple sampling repetitions are performed (e.g. 10,000 shots), the output values can be aggregated into a probability distribution. Existing testing approaches apply statistical methods to these distributions, interpreting divergences between programs as behavioral differences. For example, the widely used Kolmogorov-Smirnov test [10] assesses distributional differences to detect anomalous program behavior. In contrast, the statevector provides a mathematically rigorous representation of quantum states. Unlike measurement, statevectors do not require additional circuit gates, and their extraction does not cause quantum state collapse. A workflow comparison is illustrated in Figure 1.

We identify a major limitation in measurement-based output validation: measurement results exhibit inherent inconsistency. Since measurement results are derived from quantum circuit sampling, the observed distribution represents only an approximation of the true underlying distribution. A critical question in measurement-based testing is: How many sampling shots are sufficient to make the distribution representative of the true distribution? This question remains underexplored in the existing literature. Current approaches typically employ empirical shot numbers without rigorously assessing their sufficiency. Consequently, the inherent uncertainty in measurements undermines the reliability of downstream analyses.

Consider QDiff [15] as a representative example, where measurement results are used to determine the program's correctness. The approach executes programs repeatedly (approximately 10,000 times) to obtain sampling distributions, then applies the K-S test for statistical analysis. However, we observe that their results lack stability. When we randomly select a program and vary the sampling shot count, the results fluctuate significantly with changing sample sizes. As shown in Figure 2, the key outputs of the Kolmogorov–Smirnov test, namely the K and P values, vary considerably as the number of sampling shots increases. Since QDiff's bug detection relies on pre-defined K-value thresholds, variations in sampling shots can significantly impact bug identification decisions.

Another critical limitation of measurement-based validation is scalability. In discussions with both freshmen and senior developers from Qiskit, Cirq, and Pytket, we found a consensus that measurement-based approaches are confined to quantum program simulation. When a definite value is read out, the measurement leads to the quantum state collapse, which makes the quantum state unavailable for downstream applications. Furthermore, the measurement-based validation methods are prone to introducing inherent noise from real quantum machines, and therefore lead to a misjudgment of the quantum program's behavior. These combined drawbacks, i.e., the nature of state collapse and the vulnerability to quantum noise, create significant obstacles for reliable quality assurance in quantum program development.

B. The use of statevector-based validation

Statevector-based validation offers several advantages over measurement-based approaches: 1) Stability. Unlike measurement-based methods that rely on statistical distribution comparisons, statevectors eliminate statistical uncertainties by directly representing quantum program states without probabilistic sampling. 2) Efficiency. Statevectors are obtained through single simulation runs, whereas measurement-based approaches require thousands of executions (typically 10,000+ shots) to construct distributions. 3) Scalability. Measurement-based methods require fixed simulator seeds to ensure reproducibility, which real quantum hardware cannot provide. Statevector-based validation avoids these constraints and can readily adapt to future quantum computing environments.

However, statevector-based validation should be carefully applied in practice. After we discussed with the developers of Qiskit, we find that the global phase of a quantum circuit may affect the values of the statevector, and thus make the bit-to-bit comparison between two statevectors ineffective. To address this, we propose to use the dot product of two statevectors to help decide whether they are different or not. Specifically, if the dot product of the two statevectors is neither 1.0 (which indicates the statevectors are in the same direction) nor 0.0 (which means the statevectors are in the opposite direction), for example, 0.45, then the two statevectors can be considered as different. As shown in Figure 3, the statevector of program 1 and program 2 are different at bit-wise due to the effect of global phase. By calculating the dot product of the two statevectors, we can determine that they are the same.

$$\begin{array}{cccc} \text{Statevector of program 1} & & \text{Statevector of program 2} & \\ \left|\begin{array}{l} \texttt{0.19+0.00j, 0.19+0.00j,} \\ \texttt{0.00+0.19j, 0.00+0.19j,} \\ \texttt{0.32+0.32j, 0.32+0.32j,} \\ \texttt{-0.32+0.32j,-0.32+0.32j} \end{array}\right| & \times & \left|\begin{array}{l} \texttt{0.17-0.07j, 0.17-0.07j,} \\ \texttt{0.07+0.17j, 0.07+0.17j,} \\ \texttt{0.42+0.17j, 0.42+0.17j,} \\ \texttt{-0.17+0.42j,-0.17+0.42j} \end{array}\right| & = 1 \end{array}$$

Fig. 3: An example of using dot product for statevector-based validation.

> **Answer to RQ2:** Measurement-based validation methods are suitable for simple assessment tasks, such as determining whether specific output values appear. For complex assessment tasks, such as comparing behaviors between two programs, measurement-based approaches show significant limitations, and using the calculation of statevectors can be a good alternative.

V. Future Research

Future research could focus on developing more robust and efficient measurement strategies to address the inherent challenges posed by current quantum validation methods. Traditional measurement is limited by statistical fluctuations, loss of phase information, and hardware-induced noise, especially in real quantum devices. By improving sampling techniques, incorporating error mitigation, and designing measurement-aware validation criteria, it may be possible to extract more meaningful insights from measurement results. Furthermore, adaptive measurement protocols and hybrid classical-quantum post-processing could further enhance the reliability and informativeness of measurement-based analyses.

In parallel, the statevector is expected to play an increasingly important role in future quantum program validation. The statevector allows for fine-grained comparison, equivalence checking, and structural validation of quantum programs, particularly in simulation settings. As tools for extracting and analyzing statevectors become more standardized and efficient, statevector-based validation will likely become a critical technique for validating program behavior in quantum program testing.

VI. Conclusion

In this paper, we empirically analyze existing approaches for quantum program testing to find that measurement-based validation is largely adopted in practice. Next, we present a short discussion to compare the measurement-based validation with the statevector. Finally, we summarize the limitations of measurement-based validation and the prospects for future use of the statevector.

References

[1] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Qusbt: Search-based testing of quantum programs," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 2022, pp. 173–177.

[2] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Generating failing test suites for quantum programs with search," in *International symposium on search based software engineering*. Springer, 2021, pp. 9–25.

[3] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Application of combinatorial testing to quantum programs," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2021, pp. 179–188.

[4] X. Wang, P. Arcaini, T. Yue, and S. Ali, "Qucat: A combinatorial testing tool for quantum software," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 2066–2069.

[5] R. Abreu, J. P. Fernandes, L. Llana, and G. Tavares, "Metamorphic testing of oracle quantum programs," in *Proceedings of the 3rd International Workshop on Quantum Software Engineering*, 2022, pp. 16–23.

[6] M. Paltenghi and M. Pradel, "Morphq: Metamorphic testing of the qiskit quantum computing platform," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2413–2424.

[7] Y. Li and M. Ying, "Debugging quantum processes using monitoring measurements," *Physical Review A*, vol. 89, no. 4, p. 042338, 2014.

[8] J. Liu, G. T. Byrd, and H. Zhou, "Quantum circuits for dynamic runtime assertions in quantum computation," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1017–1030.

[9] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 541–553.

[10] K. An, "Sulla determinazione empirica di una legge didistribuzione," *Giorn Dell'inst Ital Degli Att*, vol. 4, pp. 89–91, 1933.

[11] N. V. Smirnov, "On the estimation of the discrepancy between empirical curves of distribution for two independent samples," *Bull. Math. Univ. Moscou*, vol. 2, no. 2, pp. 3–14, 1939.

[12] I. Good, "Some terminology and notation in information theory," *Proceedings of the IEE-Part C: Monographs*, vol. 103, no. 3, pp. 200–204, 1956.

[13] J. Shore and R. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Transactions on information theory*, vol. 26, no. 1, pp. 26–37, 2003.

[14] E. R. Johnston, N. Harrigan, and M. Gimeno-Segovia, *Programming Quantum Computers: essential algorithms and code samples*. O'Reilly Media, 2019.

[15] J. Wang, Q. Zhang, G. H. Xu, and M. Kim, "Qdiff: Differential testing of quantum software stacks," in *2021 36th IEEE/ACM international conference on automated software engineering (ASE)*. IEEE, 2021, pp. 692–704.

[16] I. Iwumbwe, B. Z. Liu, and J. Wickerson, "Qutefuzz: Fuzzing quantum compilers using randomly generated circuits with control flow and subcircuits," 2025.

[17] J. Wang, M. Gao, Y. Jiang, J. Lou, Y. Gao, D. Zhang, and J. Sun, "Quanfuzz: Fuzz testing of quantum program," *arXiv preprint arXiv:1810.10310*, 2018.

[18] S. Honarvar, M. R. Mousavi, and R. Nagarajan, "Property-based testing of quantum programs in q#," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 430–435.

[19] S. Xia, J. Zhao, F. Zhang, and X. Guo, "Quantum concolic testing," *Proc. ACM Softw. Eng.*, vol. 2, no. ISSTA, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3728926

[20] P. Long and J. Zhao, "Equivalence, identity, and unitarity checking in black-box testing of quantum programs," *Journal of Systems and Software*, vol. 211, p. 112000, 2024.

[21] P. Long and J. Zhao, "Testing multi-subroutine quantum programs: From unit testing to integration testing," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–61, 2024.