Quantum Squeeziness: Information-Theoretic Foundations of Quantum Software Testability

Avner Bensoussan
Department of Informatics, King's College London
London, United Kingdom
avner.bensoussan@kcl.ac.uk

Héctor Menéndez
Department of Informatics, King's College London
London, United Kingdom
hector.menendez@kcl.ac.uk

Mohammad Reza Mousavi
Department of Informatics, King's College London
London, United Kingdom
mohammad.mousavi@kcl.ac.uk

ABSTRACT

Testability—the degree to which faults can be revealed and diagnosed—has long been fundamental in software engineering but remains largely unexplored for quantum software. This work presents the first formalisation of testability for quantum programs, adapting classical concepts to quantum computation. We introduce *Quantum Squeeziness*, an information-theoretic metric inspired by classical squeeziness, to quantify how faults are masked and prevented from propagating to observable outcomes. We treat it as an inverse measure of testability: higher squeeziness implies stronger fault masking and reduced detectability.

We propose an efficient method to compute Quantum Squeeziness and validate it through large-scale quantum mutation analysis. Across more than 50k mutants, we observe a strong negative correlation between squeeziness and mutation score, showing that higher squeeziness predicts lower testability. We further introduce a technique that constrains where and how perturbations are injected, reducing cost while preserving accuracy. The computation is **15% faster than a standard error-propagation baseline**, while the optimised variant, *FastSqueeziness*, reduces mean runtime from 3200 s to 189 s (**16.9× speedup**) with **95% accuracy**.

By establishing a rigorous foundation of testability for quantum software, this work opens new directions at the intersection of software engineering, quantum information theory, and testing. Beyond testing and debugging, it also supports applications in quantum error mitigation and correction, enabling principled assessment and improvement of program robustness.

ACM Reference Format:
Avner Bensoussan, Héctor Menéndez, and Mohammad Reza Mousavi. 2026. Quantum Squeeziness: Information-Theoretic Foundations of Quantum Software Testability. In *TODO: Proceedings title.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/0000000.0000000

1 INTRODUCTION

As quantum computing moves toward practical applicability, the correctness and reliability of quantum programs have become increasingly critical [1–3]. Despite substantial progress in quantum algorithms, compilers, and hardware, the fundamental software engineering concept of testability—the extent to which faults can be revealed and diagnosed—remains largely unexplored in the quantum domain. Quantum software must not only deliver correct output, but also remain robust and predictable in the face of hardware noise, decoherence, and subtle quantum-specific faults such as phase flips or unintended entanglement. Without sufficiently high testability, faults may remain hidden, undermining confidence in both correctness and long-term reliability. Indeed, in classical software testing, testability is widely recognised as foundational to fault detection, maintainability, and software quality [4, 5]. In the quantum domain, classical notions of testability have no direct translation. Superposition, entanglement, and probabilistic measurement fundamentally reshape program behaviour and observability, while the potential exponential growth of state space means that small errors can propagate in counterintuitive ways. Observable outcomes may therefore not sensitively reflect internal state discrepancies, motivating the need for quantum-aware testability frameworks rather than naïvely re-using classical approaches [6, 7].

To address this gap, we propose a quantum-native formalism for testability, grounded in information-theoretic foundations and inspired by classical testability metrics. Central to our formulation is the concept of squeeziness [8]: the degree to which information about a fault in a quantum state is lost or "squeezed" through the circuit, making it difficult to observe in standard output statistics. Our definition is based on comparing quantum states, and we start with measuring that using observable outputs. By modelling how faults propagate through quantum states and how they influence (or fail to influence) the final state, our approach provides a principled means for assessing testability with respect to failure and error propagation. This allows us to quantify the *testability* of a quantum program, i.e., how reliably faults manifest as observable deviations rather than being masked by quantum effects. We provide a more efficient technique to confine the scope in which faults are injected and evaluated for testability. Further improvements may be achieved by incorporating measurement-based methods such as classical shadows [9, 10].

Beyond its conceptual contribution, Quantum Squeeziness is designed for automation. It can support mutant selection, test-effort prioritisation, oracle-strength selection, and fitness-guided fuzzing or circuit generation. Because it requires no human judgement and operates directly on circuit behaviour, it provides an operational notion of testability for quantum software under probabilistic execution and noise.

TODO-VENUE, TODO: dates, TODO: location

ACM ISBN 978-0-0000-0000-0/26/00
https://doi.org/10.1145/0000000.0000000

Research Questions. To bridge the gap between established software testing principles and the emerging needs of quantum computing, this study examines how the concept of testability can be meaningfully adapted to quantum programs. Building on classical foundations while accounting for the unique behaviours of quantum systems, we investigate how faults propagate through quantum states, how such propagation can be quantified, and what these metrics imply in terms of test effort and the effectiveness of fault detection. Guided by this goal, we structure our inquiry around the following research questions.

- **RQ1. *Definitions.*** How can testability and squeeziness be formally defined for quantum software systems, and what validation requirements must such definitions satisfy?
  This question investigates classical notions of testability and squeeziness and their foundations in information theory, to extend these concepts to quantum software. It further aims to define the necessary validation criteria for quantum software testability.
- **RQ2. *Metrics.*** How can testability and squeeziness be quantitatively measured in quantum software?
  This question develops metrics for both *quantum software testability* and *Quantum Squeeziness*, evaluates their interrelation, and compares their efficiency and effectiveness. It also investigates whether Quantum Squeeziness exhibits an inverse correlation with testability, as observed in classical software [8].
- **RQ3. *Implications for Testing.*** How does squeeziness relate to testing?
  This question investigates the practical applications of Quantum Squeeziness, particularly its impact on the effectiveness of testing oracles through its correlation with mutation analysis, and its potential to guide the selection and prioritisation of more testable quantum circuits.

This study offers three main novel contributions.

- We **transfer the notion of testability to a Quantum context** and present a **first metric** to assess it.
- We propose a **quantum counterpart of squeeziness**, an information-theoretic measure of **software testability**, and evaluate its **correlation with quantum testability**.
- We **validate our metrics over mutation analysis** to ensure its relevance to quantum software testing.

Structure of the paper. Section 2 introduces quantum computing, and key concepts to understand testability and its use cases in Software Engineering. In Section 3, we review the related work, both in classical and quantum software engineering. Building on the two previous sections, Section 4 provides an intuition as well as a set of requirements for quantum testability. Section 5 describes the methodology of this study, and the experimental setup we followed. In Section 6, we present our results before discussing them in Section 7. We state threats to the validity of our study in Section 8, suggest some avenues for future work in Section 9 and conclude in Section 10.

2 BACKGROUND

In this section, we briefly introduce quantum computing and key elements to understand testability and fault masking.

2.1 Quantum Computing

Quantum computing (QC) was conceptualised to simulate quantum mechanics with computers *"built of quantum mechanical elements which obey quantum mechanical laws"* [11]. It was later found to have several potential applications and could offer significant speed-up over its classical counterpart. In 1994, Shor's proposal of a polynomial-time algorithm for prime factorisation and discrete logarithms on a Quantum Computer [12] raised enormous interest due to the threat it could constitute to modern RSA cryptosystems. Soon after, Grover proposed a fast database search on quantum computers [13] that promised quadratic speed-up over the best classical algorithm. The resulting potential speed-up is often referred to as "quantum supremacy" [14]. However, demonstrating quantum supremacy on real hardware remains a long-standing challenge. In the near term, the focus is on achieving quantum advantage—showing that quantum devices can outperform classical computers on specific, practical tasks. As Preskill notes [15], this marks the first major milestone for quantum computing.

2.2 Quantum Circuits

A quantum circuit typically begins with the initialisation of qubits and ends with their measurement, where the information is translated into classical bits. The purpose of a quantum circuit is to harness the principles of quantum mechanics, such as superposition and entanglement, to perform computational tasks that would be significantly more efficient than classical computers. A qubit in superposition is represented as a linear combination of the computational basis states,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Unlike a classical bit, which must occupy exactly one state at any given time, a qubit exists in a coherent combination of both basis states prior to measurement. The squared magnitudes $|\alpha|^2$ and $|\beta|^2$ correspond to the probabilities of obtaining $|0\rangle$ or $|1\rangle$, respectively, upon measurement. Quantum circuits exploit this property to process a multitude of possibilities simultaneously, enabling quantum parallelism [16]. Quantum circuits can create entangled states where the properties of one qubit are dependent on the properties of another, even if they are physically separated. This entanglement allows for new information processing using quantum parallelism [16].

Figure 1 represents a simple four-qubit quantum circuit that is designed to produce an equally balanced distribution of all outputs. Input variables q[0] to q[3] represent the 4 input qubits. The line represents time. The square and round elements are quantum logic gates, and the final boxes elements with a needle inside represent measurements, where the qubit is observed and its state is extracted into classical bits. The gate depth of a quantum circuit measures its computational complexity as the number of sequential layers of quantum gates required from input to measurement [16]. Figure 1 represents a circuit of depth 4.

State-level representation and fault abstraction. Any quantum circuit C acting on an initial state ρ_0 prepares a quantum state $\rho = U\rho_0 U^\dagger$, where U is the unitary implemented by the circuit; for pure states, $\rho = |\psi\rangle\langle\psi|$. More generally, any physically realizable quantum process is described by a completely positive

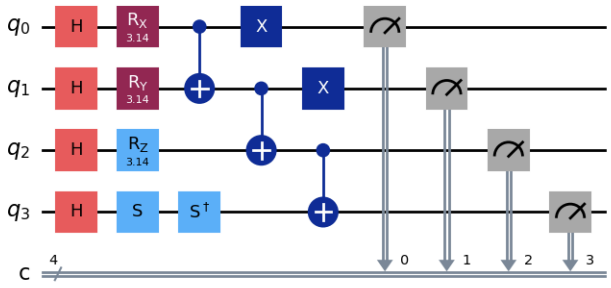

Figure 1: Simple quantum circuit creating an equally balanced distribution of all outputs.

trace-preserving (CPTP) map, i.e., a quantum channel acting on density operators [16, 17]. By the Church–Turing–Deutsch principle [18], any implementable operation admits such a representation. Modelling perturbations at the state (or channel) level therefore captures the most general semantic effect of faults, independent of their syntactic origin in the circuit, whether developer-introduced (e.g., incorrect gates or parameters) or hardware-induced, and thus forms the basis of our approach in this paper.

Similarity metrics. A quantum state can be represented either as a state vector or, more generally, as a *density matrix* ρ, which captures both pure states and statistical mixtures. Comparing quantum states therefore reduces to comparing their corresponding matrices. Two widely used metrics for this purpose are ***quantum fidelity*** and the ***quantum Hellinger distance***, both of which quantify how similar two quantum states are [16].

The *quantum fidelity* between two density matrices ρ and σ is defined as [19]:

$$F(\rho,\sigma) = \left(Tr\sqrt{\sqrt{\rho}\,\sigma\,\sqrt{\rho}}\right)^2, \tag{1}$$

where ρ and σ are density operators and Tr denotes the trace. It measures how closely the two states resemble one another, with $F(\rho,\sigma) = 1$ if and only if $\rho = \sigma$, and values closer to 0 indicating increasingly dissimilar states.

Complementarily, the *quantum Hellinger distance* [20] provides a notion of distance between states:

$$H_Q(\rho,\sigma) = \frac{1}{\sqrt{2}}\left\|\sqrt{\rho} - \sqrt{\sigma}\right\|_{\mathrm{F}}, \tag{2}$$

where $\|A\|_{\mathrm{F}} = \sqrt{\mathrm{Tr}(A^\dagger A)}$ denotes the Frobenius norm. While fidelity measures similarity, the Hellinger distance quantifies dissimilarity, taking the value 0 when the states are identical and increasing as they diverge. The two metrics are directly related by

$$H_Q(\rho,\sigma) = \sqrt{1 - \sqrt{F(\rho,\sigma)}}, \tag{3}$$

highlighting their shared geometric interpretation in the space of quantum states [16, 19].

2.3 Testability in Software Engineering

Testability. refers to the degree to which a system or component supports testing efforts, influencing how efficiently one can identify faults and verify correct functionality. Several definitions of testability exist in the literature. Both the IEEE and ISO standards relate testability to the ease of establishing the test criterion, and the performance of test to assess whether the criteria were met [21, 22]. Testability is associated with two key notions according to Binder [4]: controllability and observability. One must be able to control the input and observe the output of a system to properly test it. Voas et al. [23] focus on the sensitivity to faults. They relate testability to a notion of information loss between the input and output domains by looking at software dynamics, not just syntax. Finally, Briand et al. [24] define testability as the extent to which a module exposes information that facilitates the automatic generation of test cases. We adopt the approach of Voas et al. [23] and adapt it to quantum software.

Fault Propagation. A ***fault*** is a defect or flaw in a system component that can lead to incorrect behavior or outputs. Faults can arise due to design flaws, manufacturing defects, environmental factors, or operational developer mistakes.

Failed Error Propagation (FEP) refers to situations where an error, originating from a fault, does not propagate to the output, making it undetectable by typical testing methods [25]. This lack of error propagation can lead to "silent" errors, where faults go unnoticed by error detection mechanisms, thus impacting the reliability assessment of the system. FEP directly undermines observability: if a fault does not propagate into an error, then the observable outputs appear correct even in the presence of a latent fault. The study of error propagation has been a widely recognised technique to assess the testability of a SUT since the early 90s [26]. FEP analysis through fault injection studies is therefore a common way of assessing the testability of a system [27].

Squeeziness. is defined as a measure of the information (entropy) loss during program execution. For a deterministic program that maps inputs to outputs, let I denote the random variable over the input domain and O the induced random variable over the output domain. Then, the squeeziness is given by:

$$Sq(I,O) = H(I) - H(O) \tag{4}$$

where $H(X)$ is the Shannon entropy of the random variable X, defined as follows [28]:

$$H(X) = -\sum_{x\in X} p(x)\log p(x), \tag{5}$$

where X is a discrete random variable and $p(x)$ is the probability of each outcome x. This measure quantifies the uncertainty or information content inherent to the probability distribution of X.

The extremes of squeeziness are illustrated by the uniform and Dirac cases. If the input I is uniformly distributed over n values, then $H(I) = \log n$. If the program maps every input to a single output value (a Dirac distribution), then $H(O) = 0$, and therefore $Sq(I,O) = \log n$, which is the maximum possible information loss. Conversely, if I is itself a Dirac distribution, then $H(I) = 0$ and, since the program is deterministic, $H(O) = 0$, giving $Sq(I,O) = 0$, the minimum possible information loss. For intuition, consider a

simple classical function: if it always returns the same output, all information about the input is lost (high squeeziness), while if each input maps uniquely to an output, the input information is fully preserved (low squeeziness).

Squeeziness was proven to be efficient in several testing contexts, noticeably to assess Failed Error Propagation [29], or to detect fault-masking [30]. Androutsopoulos et al. [25] studied the relationship between Conditional Entropy and Failed Error Propagation in software testing, highlighting once more the close correlation between information theory and testability. Although quantum information theory is a very active field of research [31], to our knowledge, it has not yet been utilised in the context of quantum software testability.

3 RELATED WORK

In this section we first present the state-of-the-art in quantum software testing, introduce how information theory is used to enhance classical software testability, and then describe how testability serves in establishing software design guidance.

3.1 Testing Quantum Programs

Quantum software testing is challenging, with multiple studies highlighting the critical need for specialised testing and debugging tools for quantum programs [1, 32–35]. Several testing approaches have been explored, including property-based testing [36, 37], mutation-based [38–40], statistical assertion [41–43], search-based [44] and metamorphic testing [45] techniques. Complementing these efforts, researchers have developed quantum bug benchmarks [46–48] and identified specific bug patterns, notably in Qiskit programs [49]. MQTBench suite [50] provides a quantum circuit benchmark to support quantum software research, which we used in this study. Together, these efforts highlight the progress and gaps in quantum software testing, motivating our approach, which focuses on quantifying program testability and providing a principled method to identify circuits where faults are most likely to manifest.

3.2 Fault masking in logic and reversible circuits

Fault masking prevents faults from propagating to observable outputs, typically using redundancy or error-correcting schemes such as triple modular redundancy (TMR) or error-correcting codes (ECC) in classical circuits [51–53]. Zamani et al. [54] extended these ideas to reversible circuits, linking them to quantum computing and quantum error correction. These techniques provide intuition for quantum circuits, where superposition and interference can similarly mask faults, reducing their observability. Our work builds on this insight by quantifying fault propagation and masking to provide a principled measure of quantum program testability.

3.3 Information theory for software testability

Testability of classical software has been an active area of research for decades. In 1995, Voas et al. [23] linked testability to a notion of implicit information loss. They introduced a Domain to Range Ratio (DRR) that assesses the information loss between the input and output distribution, and allows for assessment of the likelihood of fault propagation in software. Later, Clark et al. [8] introduced an information-theoretic measure of testability, Squeeziness, that performs more efficiently than DRR.

3.4 Testability for software design guidance

Testability is not only a valuable asset to guide software testing, but its analysis can also lead to general guidelines to develop safer and more stable software. Binder [4] suggested that testability should be considered before development, and motivate design choices to maximise controllability and observability. As systems' complexity increases, testing schemes become more expensive, and designing for optimal testability becomes crucial. Woodward et al. [55] evaluate the relationships between testability, Fault size, and Domain-to-Range Ratio, suggesting that the balance between these three notions should be considered when designing software. Voas et al. [23] conclude their work with *'[...] we are convinced that this inherent software characteristic* [testability] *will become an important factor to consider during software development and assessment.'*. Similarly, we believe defining testability in a quantum software context is a crucial step towards not only better quantum testing techniques but also towards guidance to more stable quantum software design.

3.5 Positioning our study

The related work in Table 1 highlights that existing quantum software engineering approaches typically address only isolated aspects of our problem space. Prior quantum-focused tools such as MorphQ [32] and Muskit [38] provide algorithm-level measurements and practical tool support, whereas statistical assertion frameworks [41, 42] and bug benchmarks [46, 47] cover narrower evaluation needs without formal definitions or information-theoretic foundations. In contrast, classical software-engineering measures—such as DRR [55], squeeziness [8], and entropy-based analyses [25, 29, 30]—offer formal definitions and leverage information theory, but operate exclusively in classical domains and lack quantum applicability. Our work uniquely integrates all four pillars: a formal definition of testability, its efficient algorithmic measurement, quantum-native information-theoretic grounding, and an implemented tool, thereby filling a gap addressed by neither prior quantum nor classical approaches.

4 QUANTUM TESTABILITY: INTUITION AND REQUIREMENTS

While classical testability is well defined, quantum software lacks a native, formal notion. Building on the concepts introduced in Sections 2 and 3, we outline the intuition and requirements for *quantum testability*.

4.1 Intuition

A *testable quantum circuit* should allow faults to propagate to observable outputs, directly reflecting classical notions of *observability* and *sensitivity to fault* [4, 23]. Conversely, if a perturbation does not affect the output, the circuit exhibits a quantum analogue of *Failed Error Propagation (FEP)* [25], thus undermining testability.

However, unlike classical programs, internal quantum states cannot be inspected. As a result, testability must be inferred from how differences in the quantum state manifest at the output - here

Table 1: Positioning of our work with respect to related research.

Work	Application Domain	Formal Definition	Automated Measurement	Information-Theoretic Foundation	Software Implementation
Quantum-Focused Testing Techniques [32, 38]	Q	–	✓	–	✓
Statistical Assertions [41, 42]	Q	–	✓	–	–
Bug Benchmarks [46, 47]	Q	–	–	–	✓
QASMBench [56]	Q	–	–	–	✓
DRR [23]	C	✓	✓	–	–
Squeeziness [8]	C	✓	✓	✓	–
Entropy-Based [25, 29, 30]	C	✓	✓	✓	–
Our Work	**Q**	✓	✓	✓	✓

the final state vector, consistent with information-loss-based views of testability [29, 30].

Scalable and state-grounded assessment of testability. Testability is inherently a property of quantum states, but directly accessing state vectors is infeasible for large circuits. In this work, we define and analyze testability in a fully controlled, state-vector setting, where perturbations can be precisely applied and their effects directly observed. This controlled approach establishes the fundamental notions and formal properties of testability [4, 25, 28]. At the same time, our methodology is designed to be scalable: practical proxies based on weak or mid-circuit measurements and classical-shadow techniques [9, 10] can extend testability assessment to larger circuits, including those relevant for the fault-tolerant quantum computing era, without requiring full state access.

4.2 Requirements on a Quantum Testability Spectrum

We posit a natural *testability spectrum*, defined by how much an injected perturbation is preserved in the output — a direct analogue of classical FEP and *squeeziness*-based reasoning (Eq. 4) [25, 29].

At one extreme, *Constant Circuits* erase input distinctions, producing maximal information loss and strong fault masking. At the other, *Identity Circuits* preserve all deviations, maximising sensitivity and observability [23]. Between these lie general quantum programs that partially suppress or propagate perturbations.

We therefore set the following requirements:

(1) Weak propagation of perturbations implies low testability.
(2) Strong propagation of perturbations implies high testability.
(3) Quantum testability is fundamentally tied to information retention.

In Section 5, this intuition is operationalised using state-level perturbation injection and systematic comparison of the resulting output states.

4.3 Formal Definitions

Let C be a quantum circuit acting on n qubits, and let $\mathcal{E}_C$ denote the induced completely positive trace-preserving (CPTP) map [57]. Given an initial state ρ_0, the nominal output of the circuit is

$$\rho = \mathcal{E}_C(\rho_0).$$

To analyze fault propagation, we introduce a *circuit cut* at layer k, decomposing the circuit into a *prefix* channel $\mathcal{E}_{C_{0\to k}}$ and a *suffix* channel $\mathcal{E}_{C_{k\to end}}$. The intermediate state at the cut is

$$\rho_k = \mathcal{E}_{C_{0\to k}}(\rho_0),$$

and the final output can equivalently be written as

$$\rho = \mathcal{E}_{C_{k\to end}}(\rho_k).$$

A controlled perturbation of magnitude δ applied at the cut is modeled as a CPTP map $\mathcal{F}_{k,\delta}$, producing the perturbed output

$$\rho^{(k,\delta)} = \mathcal{E}_{C_{k\to end}}\Big(\mathcal{F}_{k,\delta}(\rho_k)\Big).$$

Let $d(\cdot,\cdot)$ be a quantum distance measure (e.g., Hellinger distance). The notion of quantum testability is formally defined as follows:

Definition 1 (Quantum Testability). The quantum testability of C is the expected observable deviation under controlled perturbations:

$$T(C) = \mathbb{E}_{k,\delta}\Big[d\Big(\rho, \rho^{(k,\delta)}\Big)\Big].$$

High $T(C)$ indicates strong fault propagation, while low $T(C)$ indicates fault masking.

Here, $\mathbb{E}_{k,\delta}$ averages over cut locations k and perturbation magnitudes δ. Quantum Squeeziness (Definition 2) measures how much information about injected faults is lost, capturing the circuit's tendency to mask or reveal perturbations.

Definition 2 (Quantum Squeeziness). The circuit can also be viewed as an information channel mapping perturbation magnitudes X to observable deviations Y. Let $H(\cdot)$ denote Shannon entropy. Quantum Squeeziness is defined as

$$QS(C) = H(X) - H(Y) = H(X \mid Y),$$

quantifying the information about injected perturbations that is lost during computation.

4.4 Example: Quantum Squeeziness in Simple Circuits

To illustrate Quantum Testability and Squeeziness, we consider simple single–qubit circuits and analyze how perturbations propagate to the output.

Identity Circuit. Consider

$$|0\rangle \to I \to \text{Measure},$$

with a perturbation $R_x(\delta)$ applied (representing the error in quantum state due to a fault):

$$|0\rangle \to R_x(\delta) \to I \to \text{Measure}.$$

The perturbation directly changes the measurement probabilities, so the disturbance is observable and the circuit exhibits *low Quantum Squeeziness*. Formally, taking the cut before measurement, $\rho_k = |0\rangle\langle 0|$ and $\rho^{(k,\delta)} = R_x(\delta)\rho_k R_x^\dagger(\delta)$. By Definition 1, $T(C) = d(\rho, \rho^{(k,\delta)}) \propto \delta$, and by Definition 2, $QS(C) \approx 0$, confirming high testability and low squeeziness. Such circuits are easier to test because faults propagate clearly to the measurement.

Perturbation Masking Circuit. Now consider a semantically equivalent circuit, where H denotes the Hadamard Gate:

$$|0\rangle \rightarrow H \rightarrow H \rightarrow \text{Measure}.$$

Since $HH = I$, the circuit behaves as an identity transformation in the fault-free case. If the same perturbation is inserted between the two gates,

$$|0\rangle \rightarrow H \rightarrow R_x(\delta) \rightarrow H \rightarrow \text{Measure},$$

the first Hadamard spreads the state into a superposition. When the perturbation acts in this basis, its effect becomes distributed across amplitudes, and the second Hadamard recombines them through interference. This recombination tends to reduce the observable impact of the perturbation, partially masking the fault.

Less information about the disturbance therefore reaches the output, meaning the circuit exhibits *higher Quantum Squeeziness* and reduced fault observability.

Implications for Testing. Quantum Squeeziness therefore captures how much information about injected disturbances survives until measurement: circuits with low squeeziness expose faults clearly, while circuits with high squeeziness compress or mask their observable effects, making testing more challenging.

5 METHODOLOGY

This section describes the methodology employed to answer our research questions.

5.1 Research Questions: Methodological Scope and Validation

We address three research questions corresponding to *definitions*, *metrics*, and *implications for testing*, detailing how each is answered and validated.

RQ1 – Definitions. **How can quantum software testability and Quantum Squeeziness be formally defined?** We formalise these notions by extending classical testability and information-theoretic principles to the quantum setting. Validation is conceptual, ensuring alignment with the intuition and testability spectrum, and by examining extreme cases such as Constant and Identity circuits.

RQ2 – Metrics. **How can quantum software testability and Quantum Squeeziness be quantitatively measured?** We introduce operational metrics based on systematic fault injection, output deviation, and information-theoretic analysis. Validation is empirical using a representative circuit benchmark, checking consistency with the testability spectrum and correlations between testability and squeeziness.

RQ3 – Implications for Testing. **How does Quantum Squeeziness relate to practical fault detection in quantum programs?** We assess this by correlating squeeziness with mutation scores across a large set of mutants, demonstrating predictive power for fault detectability and guiding circuit selection and test prioritisation.

5.2 Conceptual Foundations: Testability and Squeeziness

Body of Knowledge and Systematic Mapping of Classical Software Testability. To address RQ1, we relied on the holistic study of software testability by Garousi et al. [5], which systematically examined 208 papers published between 1982 and 2017 following rigorous inclusion criteria. Their work provides a comprehensive overview of the classical software testability body of knowledge but does not explicitly link testability to information theory.

To bridge this gap, we incorporated the influential work of Clark et al. on squeeziness [8] and expanded this literature set using a *snowballing approach* [58]. In this context, snowballing refers to iteratively extending the set of primary studies through both backward and forward reference analysis of key papers. From these sources, we extracted the core notions relevant to testability and analysed their applicability and potential challenges within the context of quantum computing.

We first attempted a naive translation of the classical Domain and Range notions [23]: Domain would be defined as all possible classical initialisation of the qubits, and Range as all possible classical output distributions. Squeeziness was computed as the difference of entropy between the domain and the range. This first attempt, refered as ShanonSqueeziness in the replication package, led to results violating the requirements set in Section 4.2.

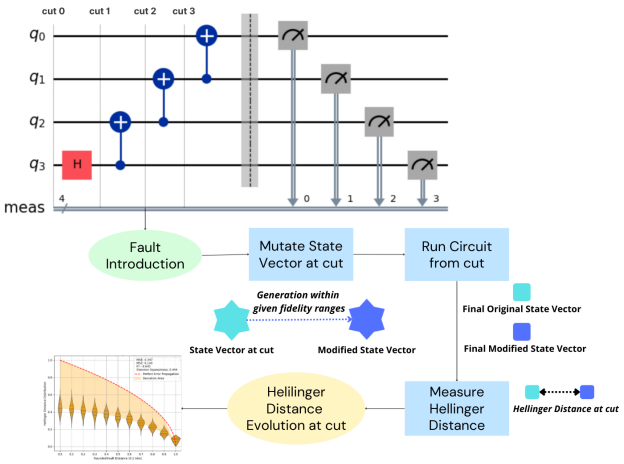

Figure 2: Pipeline for establishing the baseline of quantum circuit testability.

Our formalisation is grounded in fundamental principles of quantum software behaviour, including sensitivity to perturbations and information preservation under unitary evolution.

5.3 Metrics for Quantifying Testability and Squeeziness

To operationalise our definitions, we developed quantitative metrics for both *quantum software testability* and *Quantum Squeeziness*, enabling systematic evaluation of circuits across diverse benchmarks.

Representative Circuit Batch (Validation Dataset). To illustrate the spectrum of testability, we selected a diverse batch of quantum

circuits with known behaviours. These circuits serve as a validation dataset to assess whether the proposed metrics align with the intuitive notion of testability. We expect error propagation to increase from the least to the most testable circuit:

- **Constant Circuit:** 2 qubits; operations applied solely to the first qubit, second qubit unchanged. Qubits are swapped at the end, and only qubit 0 is measured. Disturbances are injected solely into the first qubit, leading to minimal error propagation.
- **Equally Balanced Output Circuit:** Designed to systematically output all possible outcomes with equal probability.
- **Quantum Fourier Transform (QFT) Entangled Circuit:** Produces several dominant outputs due to entanglement.
- **Amplitude Estimation Circuit:** Results in multiple dominant outputs reflecting probabilistic estimation.
- **GHZ State Circuit:** Equally entangles all qubits, producing only two equally probable states.
- **Grover Algorithm Circuit:** Oracle-based algorithm producing one dominant output through amplitude amplification.
- **Identity Implementations:** Three logically equivalent circuits producing identical final states; expected to be most sensitive to error propagation.

This circuit batch is intentionally curated to span a wide range of expected testability behaviours, rather than constituting a single benchmark suite. This design allows us to systematically probe how different structural and algorithmic characteristics influence testability across diverse scenarios. We complement this initial set with the widely used *MQTBench* [50] and *QuRaTest* [59] benchmarks to ensure broader coverage, facilitate comparison with prior work, and validate that our observations generalize beyond curated examples when addressing the remaining research questions.

Quantum Software Testability Metric. Testability is measured by systematically introducing controlled perturbations into the state vector and quantifying the resulting output deviation, as illustrated in Figure 2:

- Each circuit is cut at all gate boundaries, creating multiple evaluation points for fault injection.
- Perturbations are introduced as single-qubit or multi-qubit state deviations with fidelity ranging from 0.0 to 1.0 in steps of 0.1.
- For each perturbation, the output state is compared to the original using Hellinger distance, capturing the sensitivity of the circuit to small faults.
- The distribution of Hellinger distances across all cuts and fidelities is used to compute the circuit's overall testability score.

Quantum Squeeziness Metric. Quantum Squeeziness captures the degree to which information about injected disturbances is preserved or attenuated through the circuit. The circuit is modelled as an information channel, as shown in Figure 3:

- **Input:** Fidelity distance of the injected disturbance.
- **Output:** Hellinger distance between perturbed and original outputs.
- **Computation:** Squeeziness is quantified as the entropy difference between input and output distributions.

Using the Blahut–Arimoto algorithm, we estimated channel capacity and computed mutual information between the ideal reference channel and the channel induced by fault propagation.

FastSqueeziness. We enhanced the computation of Quantum Squeeziness to achieve higher speed while preserving accuracy. First, we evaluated all combinations of ten fidelity ranges between 0 and 1, with a step of 0.1. Second, we sequentially removed circuit cuts and assessed the resulting impact on both accuracy and computational cost. The combination of these optimisations led to a faster variant of the metric, referred to as *Fast Squeeziness*. FastSqueeziness reduces computation by selecting representative fidelity points, removing low-impact cuts, and flattening dynamic circuits. Finally, we applied Fast Squeeziness to the larger circuit benchmark from Ye et al. [59]. Since some circuits in this benchmark were dynamic, we systematically converted each dynamic circuit into all possible execution paths to enable state-vector analysis.

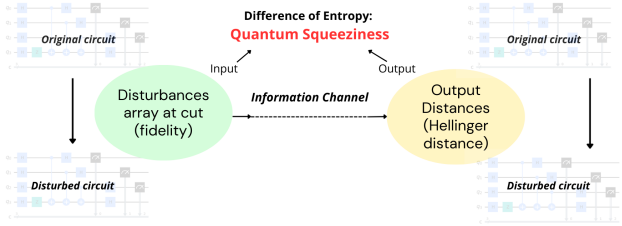

Figure 3: Pipeline to compute Quantum Squeeziness.

5.4 Fault Injection

Initially, modified state vectors were generated by repeated random sampling until the desired fidelity range was reached. For 4-qubit systems and fidelities above 0.6, this approach became impractical, often requiring millions of attempts. After testing different strategies, we found that separating the fidelity regimes was optimal: for fidelities above 0.9, the modified state is constructed geometrically as a controlled combination of the original state and a randomly generated orthogonal component, ensuring precise fidelity by construction. For lower fidelities, the current state is slightly perturbed using a near-identity unitary derived from a small random Hermitian matrix, allowing smooth and controlled exploration of nearby states. Plateau detection and occasional random resets prevent stagnation, ensuring rapid convergence. In most cases, the target fidelity is achieved in the first attempt. This fault-injection approach is not limited to classical simulation. Analogous perturbations could be implemented on quantum hardware via controlled unitary operations or noise engineering techniques, enabling experimental exploration of such faults in real quantum computations.

Why perturbations instead of input variation. Input variation conflates algorithmic behaviour with fault propagation. By injecting controlled perturbations directly into the state evolution, we isolate fault propagation and obtain a testability measure independent of the function computed by the circuit. The Shannon-Squeeziness baseline (Section 5.2) empirically illustrates the limitations of the input-variation approach.

5.5 Handling Dynamic Circuits

The QuraTest benchmark [59], used in our larger experiment, includes dynamic circuits containing mid-circuit measurements implemented via conditional operations. These circuits are incompatible with state-vector simulation. To address this, we implemented a function that systematically generates all deterministic execution paths for each dynamic circuit. Each conditional operation forces the measured qubit into either 0 or 1 and applies the corresponding sequence of gates, yielding a set of deterministic sub-circuits suitable for state-vector analysis.

5.6 Mutation-Based Validation and Mutation Score

To validate squeeziness, we used Muskit [38] to generate mutants of the selected circuits. Only non-equivalent mutants within 5% of the original circuit's squeeziness were retained to ensure comparability. For dynamic circuits with highly similar branches, we subsampled 10% of mutants per branch to avoid overrepresentation.

The mutation score is defined as the percentage of mutants detected by the test oracle. For each mutant, its output distribution is compared against the oracle using a chi-square test with a significance threshold of 5% ($p < 0.05$). A mutant is classified as detected if more than 10% of its test inputs exhibit a statistically significant deviation. The mutation score is obtained by dividing the number of detected mutants by the total number of mutants. Figure 4 shows the validation pipeline described here to answer RQ3.

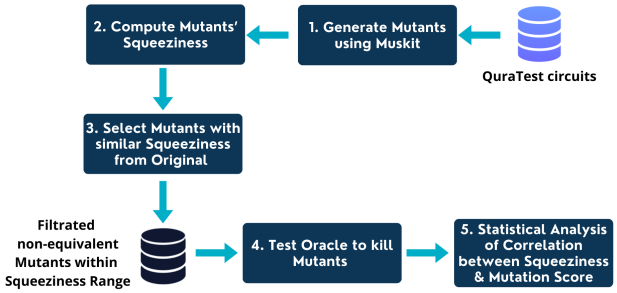

Figure 4: Validation pipeline for Quantum Squeeziness. Mutants are generated using Muskit, filtered to remain within 5% of the original circuit's squeeziness, and tested using a fidelity-based oracle. Correlations between squeeziness and mutation scores are computed to assess the metric's predictive power.

5.7 Experimental Environment and Reproducibility

All experiments were conducted using Qiskit version 1.3.2 and Python 3.9.6 to ensure optimal compatibility. Known algorithms were obtained from the MQT benchmark [50] as QASM files, while constant and identity circuits were implemented directly in QASM. For larger-scale experiments, we used circuits from Ye et al. [59]. The fully documented replication package is available at Zenodo. All random processes were executed with fixed seeds to ensure reproducibility.

6 RESULTS

In this section, we present our results, organised around each research questions.

6.1 RQ1: Definitions — Quantum software testability and Quantum Squeeziness

Classical notions of software testability rely on deterministic fault propagation and observable outputs. In quantum systems, these assumptions fail due to superposition, entanglement, and measurement-induced collapse. Faults do not propagate deterministically, and measurement yields only probabilistic information. Classical testability metrics, therefore, cannot reliably capture quantum circuit behavior. Figure 5 represents two extreme examples violating our requirements.

We define two complementary concepts:

(1) Quantum Testability: the ability of a quantum circuit to propagate injected faults to outputs so that they can be observed. This captures the intuition from Section 4: circuits that preserve deviations in the output are highly testable, while those that mask perturbations are not.
(2) Quantum Squeeziness: an information-theoretic metric quantifying the loss of information between a faulty input and the resulting output distribution. Squeeziness operationalizes the testability spectrum from Section 4, measuring how observable outputs are affected by faults.

Constant circuits suppress disturbances and show low testability with high squeeziness, whereas Identity circuits propagate all deviations and show high testability with low squeeziness. These definitions satisfy the validation requirements 1 and 2 derived from the intuition and testability spectrum in Section 4.2.

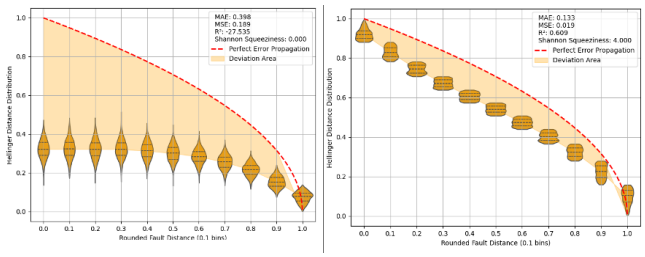

Figure 5: A Naive definition of Squeeziness violates our requirements: On the left the equally balanced circuit has high FEP but a Shanon Squeeziness of 0 (domain and range have the same entropy), and on the right the Identity circuit has very low FEP but a high Shanon Squeeziness of 4 (only one possible output leads to a high difference of entropy).

RQ1: Quantum Testability and Squeeziness

Quantum testability is the expected observable divergence caused by controlled perturbations, capturing how strongly faults propagate to the output. Quantum Squeeziness measures the entropy loss between injected disturbances and resulting deviations, quantifying information compression during computation.
Circuits that suppress perturbations exhibit low testability and high squeeziness, whereas circuits that preserve or amplify them exhibit the opposite. This formalisation avoids naive classical adaptations by accounting for superposition, measurement collapse, and probabilistic observability in quantum programs.

6.2 RQ2: Metrics — Quantifying Testability and Squeeziness

Baseline experiments. Extreme cases illustrate the spectrum of error propagation (Figure 7). Constant circuits show minimal propagation, indicating low testability, while Identity circuits propagate faults strongly, indicating high testability. Here, R^2 refers to the coefficient of determination measuring the distance between the ideal fault propagation predicted by Eq. (3) and the empirical observations, and serves as an empirical metric for assessing fault propagation. This is consisten with requirements 1 and 2 set in Sec. 4.2.

Quantum Squeeziness metric. Circuits that propagate faults efficiently have low squeeziness, while circuits that suppress faults have high squeeziness. The metric correlates inversely with testability ($Spearman = -0.85$) as indicated in Figure 6 and Table 2. This is consistent with requirement 1 set in Sec. 4.2.

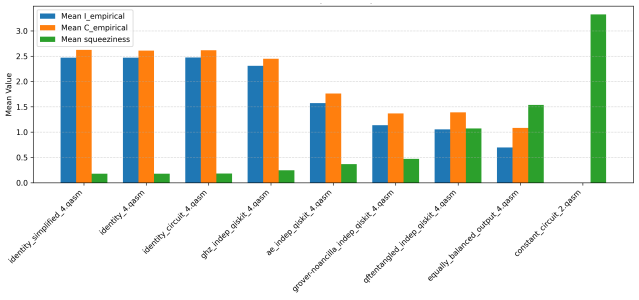

Figure 6: Quantum Squeeziness across circuits. Low squeeziness corresponds to high fault propagation (high testability).

FastSqueeziness optimisation. Across the full dataset (50,000+ mutants), the average runtime per circuit decreases from 3200 s to 189 s, corresponding to a 16.9× reduction. Despite this acceleration, result quality is preserved: FastSqueeziness maintains high accuracy (95% on average, with a standard deviation of 9.3%), and mean squeeziness remains comparable to the regular method (regular: 0.99, fast: 1.02). These results demonstrate that FastSqueeziness enables scalable evaluation without sacrificing accuracy.

Detailed circuit results. Table 2 lists Squeeziness and R^2 values, ordered from low to high squeeziness. Identity circuits have low squeeziness and strong positive correlation with testability, whereas Constant and equally balanced circuits have high squeeziness and weaker or negative correlation.

Table 2: Quantum Squeeziness and R^2 across circuits (low to high Squeeziness).

Circuit	Squeeziness	R^2
identity_simplified_4	0.174	0.605
grover-noancilla_4	0.176	0.912
identity_4	0.184	0.611
identity_circuit_4	0.181	0.608
ae_4	0.339	0.171
ghz_4	0.244	0.453
qftentangled_4	1.075	-9.471
equally_balanced_output_4	1.536	-27.739
constant_circuit_2	3.322	-1.16e+30

RQ2: Metric Properties and Efficiency

Quantum Squeeziness—measuring entropy loss between injected faults and observed deviations—is strongly and inversely correlated with testability (Spearman $\rho = -0.85$). Identity circuits show minimal squeeziness and maximal testability, while constant-output circuits exhibit the opposite, confirming theoretical expectations.
FastSqueeziness reduces computation via cut and perturbation sampling, achieving a 16.9× speedup while preserving 95% ranking accuracy, enabling scalable evaluation without exhaustive state perturbation.

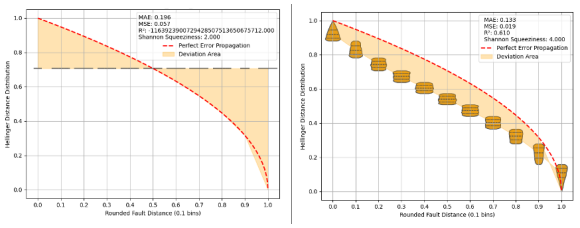

Figure 7: Extreme error propagation: Constant circuit suppresses faults (left), Identity circuit propagates faults strongly (right).

6.3 RQ.3: Implications of Testability in Testing

Figure 8 illustrates the relationship between Quantum Squeeziness and mutation score. Across more than 50,000 mutants, evaluated over three months on three parallel nodes, ***circuits with higher squeeziness consistently achieve lower mutation scores***. This trend is strongly supported by Pearson correlation ($r = -0.563$, $p = 1.54 \times 10^{-04}$), indicating that fault detection becomes significantly harder as squeeziness increases.

These results answer RQ3: *squeezy circuits are demonstrably harder to test*, as high-squeeziness circuits suppress fault propagation, whereas low-squeeziness circuits, such as identity-like structures, expose faults more readily.

To further answer RQ3, ***Quantum Squeeziness can guide circuit selection for testing***, enabling the identification of circuits where testing is most effective and highlighting those that may require stronger oracles. Overall, *high squeeziness predicts low fault detectability, while low squeeziness identifies circuits with inherently higher testability*, which alligns with its classical counterpart [8].

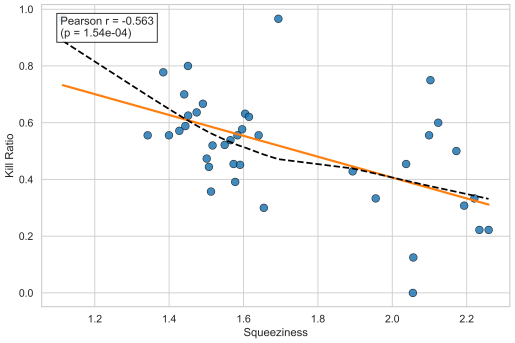

Figure 8: Correlation between Quantum Squeeziness and mutation score. High squeeziness correlates with low mutation score, indicating reduced testability.

Sensitivity to the squeeziness filter. To verify that the observed correlation is not an artefact of the 5% squeeziness filter (Section 5.6), we repeated the entire RQ3 pipeline while varying the filter threshold from 1% up to no filter at all, re-running the same mutant-selection, oracle, and correlation steps used for the headline result at each setting. Table 3 reports, for each threshold, the number of circuits retained and the resulting Pearson and Spearman correlations between squeeziness and mutation score. The negative correlation remains stable and statistically significant across all thresholds, confirming that the filter serves its intended purpose of comparing mutants within the same testability neighbourhood rather than driving the conclusion. Correlations at every threshold are computed with the identical procedure as the headline result, including its point-sampling step. The driver reproducing this table is included in the replication package (`sensitivity_table.py`).

Table 3: Sensitivity of the squeeziness–mutation-score correlation to the mutant filter threshold. The -0.563 Pearson value at the 5% setting is the headline RQ3 correlation.

Threshold	#Circuits	Pearson r	p	Spearman ρ	p
1%	278	-0.361	$2.2e-02$	-0.228	$1.6e-01$
2.5%	284	-0.044	$7.9e-01$	-0.147	$3.7e-01$
5% (used)	288	-0.563	$1.5e-04$	-0.539	$3.3e-04$
10%	288	-0.560	$1.7e-04$	-0.530	$4.3e-04$
20%	288	-0.506	$8.7e-04$	-0.470	$2.2e-03$
No filter	288	-0.248	$1.2e-01$	-0.161	$3.2e-01$

On the oracle thresholds. The mutation oracle inherits two thresholds from Muskit's defaults: the chi-square significance level ($p < 0.05$) and the detection rule (a mutant is detected if more than 10% of its test inputs deviate significantly). We adopt these default settings and rely on the empirical justification provided in the original Muskit study [38], which validated them across a range of quantum circuits; the per-input chi-square results that feed the oracle are available in the replication package for readers who wish to apply a different decision rule.

RQ3: Implications for Quantum Testing

Quantum Squeeziness is negatively correlated with mutation score ($r = -0.563$): circuits with higher information compression are harder to test, while low-squeeziness circuits expose faults more reliably.
Squeeziness therefore predicts test effectiveness and can guide circuit selection, test prioritisation, and the use of stronger statistical or oracle-based techniques where needed.

7 DISCUSSION

In this Section, we discuss the results presented earlier.

7.1 Stability of Squeeziness Across the Circuit

Our experiments indicate that the location of the injected fault in a noiseless quantum circuit has minimal impact on its propagation. We attribute this to the inherent reversibility of ideal quantum operations. Importantly, we do not expect hardware noise to invalidate the approach. Rather, we expect squeeziness to *vary* under noise: noise breaks the reversibility of the ideal evolution and alters fault-propagation behaviour, so that squeeziness becomes location- and noise-dependent. This is itself an opportunity—analysing the variation of squeeziness across cuts could expose specific sources of hardware noise, such as faulty gates or qubit decoherence, and could inform noise-mitigation mechanisms. Future work should evaluate how noise, entanglement, and circuit depth affect squeeziness stability and whether localized deviations can be systematically used for hardware diagnostics.

7.2 Squeeziness of Mutants

We observed that squeeziness responds differently to mutations depending on the circuit. While mutant generation is independent of predefined squeeziness (see Section 5 for details on the mutation process), some circuits produce only a few mutants that happen to fall within a given squeeziness range of the original circuit, whereas others allow many more. This variability suggests that certain circuits are more structurally robust or "squeeziness-stable" than others, possibly due to differences in entanglement structure, gate composition, or circuit depth. Further investigation is warranted to understand which circuit properties influence the distribution of mutants within a squeeziness threshold and how this insight could guide both mutation-based testing and quantum software design.

7.3 Error Propagation Behaviors of Circuits

Our results show that quantum circuits exhibit distinct error propagation behaviours, ranging from fault masking to effective fault

propagation. These behaviours determine testability: circuits that propagate perturbations to the output are more testable, while those that suppress them exhibit lower testability and higher Quantum Squeeziness. This allows us to link testability to structural aspects of quantum circuit design. Future work should identify sub-circuits or patterns that influence squeeziness and analyze how design choices—such as gate selection, qubit connectivity, and entanglement—affect the testability of larger circuits. As quantum software grows in complexity, testing challenges will increase, making design-for-testability considerations potentially even more critical than in classical software [23].

7.4 Implications for Quantum Software Engineering

Overall, our findings suggest that squeeziness is a promising metric for guiding quantum software testing and design. It can serve as a fast and efficient method for selecting testable mutants, assessing error propagation tendencies, and identifying structurally sensitive sub-circuits. Integrating squeeziness into quantum software development pipelines could facilitate the early detection of design weaknesses and improve overall software robustness, especially as circuits scale and hardware noise becomes a more significant factor.

8 THREATS TO VALIDITY

In this section, we discuss the potential threats to the validity of our study.

8.1 Internal Threats

The perturbation model used throughout our experiments includes a partial trace appears as an internal implementation detail. Although this introduces a fixed approximation, it does not affect the relative comparison of squeeziness across circuits. The partial-trace mechanism was chosen for its practicality in state-vector simulations, enabling precise control over fault injection and error propagation. While more optimized implementations—such as improved information-channel mechanisms—may enhance computational efficiency, the current design remains sufficient for reliable testing and benchmarking. These limitations therefore impact absolute accuracy rather than internal consistency.

8.2 External Threats

Quantum Squeeziness currently relies on simulator-level state-vector access, which cannot be reproduced on physical quantum hardware. Real devices provide only measurement outcomes, and noise affects both state evolution and readout. This limits direct applicability and poses a major threat to external validity.

A central challenge is defining or approximating the "final state vector" on hardware. Direct reconstruction is infeasible, but weak or gentle measurements, mid-circuit operations, and classical-shadow techniques offer promising alternatives. Parallel work has developed an efficient, scalable method for comparing quantum states [10, 60], which we plan to adapt for this purpose. In parallel work, we have developed a classical-shadow-based approach for scalable, hardware-compatible state comparison and live noise fingerprinting [10, 60]. It leverages the logarithmic sample complexity of classical shadows [9] to compare large quantum states from measurement outcomes alone, removing the dependence on full state-vector access, and provides a concrete path toward instantiating the squeeziness pipeline in a measurement-based setting. Developing a practical hardware-compatible formulation of squeeziness remains an important direction for future work.

9 FUTURE WORK

In this study, we introduced Quantum Squeeziness, a novel metric for quantum software testability, and demonstrated one application: selecting mutants within a defined squeeziness range. Future work will explore additional applications, including integrating squeeziness into differential testing based on fuzzing and using it as a fitness function in automated quantum circuit generation or evolutionary algorithms. We are currently developing entropy-based fuzzing approaches that use squeeziness as a fitness function to guide input and perturbation generation, with promising preliminary results.

We also plan to investigate hardware applicability by analyzing squeeziness in noisy environments, examining how variations across cuts could reveal noise or hardware errors. Furthermore, we aim to extend the notion of squeeziness to guide and optimize the design of circuits in hybrid quantum-classical architectures, potentially improving performance and testability in variational and other hybrid algorithms.

10 CONCLUSION

This work introduced the first formalisation of testability for quantum programs, establishing a foundation for analyzing how faults propagate to observable failures. By defining Quantum Squeeziness and proposing efficient computation methods, we showed that testability can be quantified in a quantum context and correlates strongly with fault detectability. Our experiments demonstrated that Quantum Squeeziness provides accurate and computationally efficient insights, with FastSqueeziness offering substantial performance gains while maintaining high fidelity.

Looking forward, we envision broader applications of squeeziness: guiding automated quantum circuit generation, serving as a fitness function in evolutionary algorithms, and informing design guidelines by identifying sub-circuits that contribute most to low testability. Evaluating squeeziness under noise could enable efficient error detection and fault localisation on real hardware. These directions highlight the potential for integrating testability analysis into quantum software development, with implications for testing, debugging, error mitigation, and circuit design.

Code and Data Availability. The fully documented replication package is archived here.

Acknowledgments. Avner Bensoussan and Mohammad Reza Mousavi have been partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/2, EPSRC project on Verified Simulation for Large Quantum Systems (VSL-Q), grant reference EP/Y005244/1 and the EPSRC project on Robust and Reliable Quantum Computing (RoaRQ), Investigation 009 Model-based monitoring and calibration of quantum computations (ModeMCQ), grant reference EP/W032635/1.

REFERENCES

[1] A. Miranskyy and L. Zhang, "On testing quantum programs," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, 2019, pp. 57–60. [Online]. Available: https://dl.acm.org/doi/10.1109/ICSE-NIER.2019.00023

[2] A. García de la Barrera, I. García-Rodríguez de Guzmán, M. Polo, and M. Piattini, "Quantum software testing: State of the art," *Journal of Software: Evolution and Process*, vol. 35, no. 4, p. e2419, 2023, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2419. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2419

[3] S. A. Metwalli and R. V. Meter, "Testing and Debugging Quantum Circuits," *IEEE Transactions on Quantum Engineering*, pp. 1–15, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10463159/

[4] R. V. Binder, "Design for testability in object-oriented systems," *Commun. ACM*, vol. 37, no. 9, pp. 87–101, 1994. [Online]. Available: https://dl.acm.org/doi/10.1145/182987.184077

[5] V. Garousi, M. Felderer, and F. N. Kılıçaslan, "A survey on software testability," *Information and Software Technology*, vol. 108, pp. 35–64, Apr. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584918302490

[6] D. Fortunato, L. Jiménez-Navajas, J. Campos, and R. Abreu, "Verification and validation of quantum software," in *Quantum Software*, I. Exman, R. Pérez-Castillo, M. Piattini, and M. Felderer, Eds. Springer, Cham, 2024, p. 93–123.

[7] N. H. Oldfield, C. Laaber, T. Yue, and S. Ali, "Faster and better quantum software testing through specification reduction and projective measurements." [Online]. Available: http://arxiv.org/abs/2405.15450

[8] D. Clark and R. M. Hierons, "Squeeziness: An information theoretic measure for avoiding fault masking," *Information Processing Letters*, vol. 112, no. 8, pp. 335–340, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002001901200021X

[9] H.-Y. Huang, R. Kueng, and J. Preskill, "Predicting Many Properties of a Quantum System from Very Few Measurements," *Nature Physics*, vol. 16, no. 10, pp. 1050–1057, Oct. 2020, arXiv:2002.08953 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2002.08953

[10] A. Bensoussan, E. Chachkarova, K. Even-Mendoza, S. Fortz, and V. Klimis, "Toward live noise fingerprinting in quantum software engineering," 2025. [Online]. Available: https://arxiv.org/abs/2512.18667

[11] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[12] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: https://epubs.siam.org/doi/10.1137/S0097539795293172

[13] L. K. Grover, "A fast quantum mechanical algorithm for database search," Nov. 1996. [Online]. Available: https://dl.acm.org/doi/10.1145/237814.237866

[14] F. A. et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, oct 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-1666-5

[15] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, arXiv:1801.00862 [cond-mat, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/1801.00862

[16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010, ISBN: 9780511976667 Publisher: Cambridge University Press. [Online]. Available: https://www.cambridge.org/highereducation/books/quantumcomputation-and-quantuminformation/01E10196D0A682A6AEFFEA52D53BE9AE

[17] M. M. Wilde, *Quantum Information Theory*, 2nd ed. Cambridge University Press, 2017.

[18] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[19] R. Jozsa, "Fidelity for Mixed Quantum States," *Journal of Modern Optics*, vol. 41, no. 12, pp. 2315–2323, Dec. 1994. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/09500349414552171

[20] E. Hellinger, "Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen." *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 136, pp. 210–271, Jul. 1909, publisher: De Gruyter. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/crll.1909.136.210/html

[21] IEEE Standards Association, "IEEE standards association," 1990. [Online]. Available: https://standards.ieee.org/ieee/610.12/855/

[22] ISO/IEC, "ISO/IEC 25002:2024 – systems and software engineering — systems and software quality requirements and evaluation (SQuaRE) — quality model overview and usage," 2024. [Online]. Available: https://www.iso.org/standard/78175.html

[23] J. Voas and K. Miller, "Software testability: the new verification," *IEEE Software*, vol. 12, no. 3, pp. 17–28, 1995, conference Name: IEEE Software. [Online]. Available: https://ieeexplore.ieee.org/document/382180/?arnumber=382180

[24] L. C. Briand, Y. Labiche, and H. Sun, "Investigating the use of analysis contracts to improve the testability of object-oriented code," *Software: Practice and Experience*, vol. 33, no. 7, pp. 637–672, 2003, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.520. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.520

[25] K. Androutsopoulos, D. Clark, H. Dan, R. M. Hierons, and M. Harman, "An analysis of the relationship between conditional entropy and failed error propagation in software testing," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. Association for Computing Machinery, 2014, pp. 573–583. [Online]. Available: https://dl.acm.org/doi/10.1145/2568225.2568314

[26] K. Miller, L. Morell, R. Noonan, S. Park, D. Nicol, B. Murrill, and M. Voas, "Estimating the probability of failure when testing reveals no failures," *IEEE Transactions on Software Engineering*, vol. 18, no. 1, pp. 33–43, 1992.

[27] Z. Li, H. Menon, K. Mohror, P.-T. Bremer, Y. Livant, and V. Pascucci, "Understanding a program's resiliency through error propagation," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. Virtual Event Republic of Korea: ACM, Feb. 2021, pp. 362–373. [Online]. Available: https://dl.acm.org/doi/10.1145/3437801.3441589

[28] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x

[29] A. Ibias, R. M. Hierons, and M. Núñez, "Using squeeziness to test component-based systems defined as finite state machines," *Information and Software Technology*, vol. 112, pp. 132–147, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919300953

[30] K. Patel, R. M. Hierons, and D. Clark, "An information theoretic notion of software testability," *Information and Software Technology*, vol. 143, p. 106759, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921002056

[31] C. Bennett, "Quantum information theory," *IEEE Transactions on Information Theory*, 1998. [Online]. Available: https://www.academia.edu/3729794/Quantum_Information_Theory

[32] M. Paltenghi and M. Pradel, "MorphQ: Metamorphic Testing of the Qiskit Quantum Computing Platform ," in *IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, ser. ICSE '23. Melbourne, Victoria, Australia: IEEE Press, 2023, p. 2413–2424. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00202

[33] ——, "Bugs in Quantum computing platforms: an empirical study," *Proceedings of the ACM on Programming Languages*, vol. 6, no. OOPSLA1, pp. 1–27, Apr. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3527330

[34] S. A. Metwalli and R. Van Meter, "A Tool For Debugging Quantum Circuits," May 2022. [Online]. Available: https://www.computer.org/csdl/proceedings-article/qce/2022/911300a624/1IvLZRlty80

[35] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Software: Practice and Experience*, vol. 52, no. 1, pp. 66–114, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/spe.3039

[36] S. Honarvar, M. R. Mousavi, and R. Nagarajan, "Property-based testing of quantum programs in q#," *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219977843

[37] G. J. Pontolillo, M. R. Mousavi, and M. Grzesiuk, "Qucheck: A property-based testing framework for quantum programs in qiskit," 2025, arXiv preprint.

[38] E. Mendiluze, S. Ali, P. Arcaini, and T. Yue, "Muskit: A mutation analysis tool for quantum software testing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Los Alamitos, CA, USA: IEEE, 2021, pp. 1266–1270, ISSN: 2643-1572. [Online]. Available: https://ieeexplore.ieee.org/document/9678563

[39] X. Wang, T. Yu, P. Arcaini, T. Yue, and S. Ali, "Mutation-based test generation for quantum programs with multi-objective search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1345–1353. [Online]. Available: https://doi.org/10.1145/3512290.3528869

[40] D. Fortunato, J. Campos, and R. Abreu, "Mutation testing of quantum programs written in QISKit," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 358–359. [Online]. Available: https://doi.org/10.1145/3510454.3528649

[41] 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*, Jun. 2019, pp. 541–553, arXiv:1905.09721 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1905.09721

[42] G. Li, L. Zhou, N. Yu, Y. Ding, M. Ying, and Y. Xie, "Proq: Projection-based Runtime Assertions for Debugging on a Quantum Computer," May 2020,

arXiv:1911.12855 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1911.12855

[43] C. G. Kang, J. Lee, and H. Oh, "Statistical testing of quantum programs via fixed-point amplitude amplification," *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA2, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3689716

[44] 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*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 173–177. [Online]. Available: https://doi.org/10.1145/3510454.3516839

[45] R. Abreu, J. P. Fernandes, L. Llana, and G. Tavares, "Metamorphic Testing of Oracle Quantum Programs," in *2022 IEEE/ACM 3rd International Workshop on Quantum Software Engineering (Q-SE)*, May 2022, pp. 16–23. [Online]. Available: https://ieeexplore.ieee.org/document/9814906

[46] P. Zhao, J. Zhao, Z. Miao, and S. Lan, "Bugs4Q: A Benchmark of Real Bugs for Quantum Programs," Sep. 2021, arXiv:2108.09744 [cs]. [Online]. Available: http://arxiv.org/abs/2108.09744

[47] J. Campos and A. Souto, "QBugs: A Collection of Reproducible Bugs in Quantum Algorithms and a Supporting Infrastructure to Enable Controlled Quantum Software Testing and Debugging Experiments," Mar. 2021, arXiv:2103.16968 [cs]. [Online]. Available: http://arxiv.org/abs/2103.16968

[48] J. Luo, P. Zhao, Z. Miao, S. Lan, and J. Zhao, "A comprehensive study of bug fixes in quantum programs," 2022. [Online]. Available: https://www.computer.org/csdl/proceedings-article/saner/2022/378600b239/1FbT6n3hGaA

[49] P. Zhao, J. Zhao, and L. Ma, " Identifying Bug Patterns in Quantum Programs ," Los Alamitos, CA, USA, pp. 16–21, Jun. 2021. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/Q-SE52541.2021.00011

[50] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing," *Quantum*, 2023, MQT Bench is available at https://www.cda.cit.tum.de/mqtbench/.

[51] F. Dias, "Fault masking in combinational logic circuits," *IEEE Transactions on Computers*, vol. C-24, no. 5, pp. 476–482, 1975, conference Name: IEEE Transactions on Computers. [Online]. Available: https://ieeexplore.ieee.org/document/1672842/?arnumber=1672842

[52] N. K. Jha, "Fault-tolerant computer system design," *IEEE Parallel & Distributed Technology: Systems & Applications*, vol. 4, no. 4, pp. 84–84, 1996, conference Name: IEEE Parallel & Distributed Technology: Systems & Applications. [Online]. Available: https://ieeexplore.ieee.org/document/7102341

[53] B. W. Johnson, *Design and Analysis of Fault-tolerant Digital Systems*. Addison-Wesley Publishing Company, 1989.

[54] M. Zamani, N. Farazmand, and M. B. Tahoori, "Fault masking and diagnosis in reversible circuits," in *2011 Sixteenth IEEE European Test Symposium*, 2011, pp. 69–74, ISSN: 1558-1780. [Online]. Available: https://ieeexplore.ieee.org/document/5957925/

[55] M. Woodward and Z. Al-Khanjari, *Testability, fault size and the domain-to-range ratio: An eternal triangle*. ACM Press, 2000, vol. 25.

[56] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "QASMBench: A low-level quantum benchmark suite for NISQ evaluation and simulation," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, pp. 10:1–10:26, 2023. [Online]. Available: https://doi.org/10.1145/3550488

[57] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge: Cambridge University Press, 2010.

[58] J. Hirt, T. Nordhausen, T. Fuerst, H. Ewald, and C. Appenzeller-Herzog, "Guidance on terminology, application, and reporting of citation searching: the TARCiS statement," *BMJ*, vol. 385, p. e078384, May 2024, publisher: British Medical Journal Publishing Group Section: Research Methods & Reporting. [Online]. Available: https://www.bmj.com/content/385/bmj-2023-078384

[59] J. Ye, S. Xia, F. Zhang, P. Arcaini, L. Ma, J. Zhao, and F. Ishikawa, "QuraTest: Integrating quantum specific features in quantum program testing," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '23. Los Alamitos, CA, USA: IEEE Press, 2024, p. 1149–1161. [Online]. Available: https://doi.org/10.1109/ASE56229.2023.00196

[60] V. Klimis, A. Bensoussan, E. Chachkarova, K. Even-Mendoza, S. Fortz, and C. Lenihan, "Shaking up quantum simulators with fuzzing and rigour," in *Proceedings of the ACM on Programming Languages*, vol. 9, no. OOPSLA2. ACM, 2025, pp. 1400–1428. [Online]. Available: https://dl.acm.org/doi/10.1145/3763100