Gleipnir: Toward Practical Error Analysis for Quantum Programs

paper #27

Abstract

Practical error analysis is essential for the design, optimization, and evaluation of Noisy Intermediate-Scale Quantum (NISQ) computing. However, bounding errors in quantum programs is a grand challenge, because the effects of quantum errors depend on exponentially large quantum states. In this work, we present Gleipnir, a novel methodology toward practically computing verified error bounds in quantum programs. Gleipnir introduces the $(\hat{\rho}, \delta)$-diamond norm, an error metric constrained by a quantum predicate consisting of the approximate state $\hat{\rho}$ and its distance δ to the ideal state ρ. This predicate $(\hat{\rho}, \delta)$ can be computed *adaptively* using tensor networks based on *Matrix Product States* (MPS). Gleipnir features a lightweight logic for reasoning about error bounds in noisy quantum programs, based on the $(\hat{\rho}, \delta)$-diamond norm metric. Our experimental results show that Gleipnir is able to efficiently generate tight error bounds for real-world quantum programs with 10 to 100 qubits, and can be used to evaluate the error mitigation performance of quantum compiler transformations.

1 Introduction

Recent quantum supremacy experiments [4] have heralded the Noisy Intermediate-Scale Quantum (NISQ) era [40], where noisy quantum computers with 50-100 qubits are used to achieve tangible performance gains over classical computers. While this goal is promising, there remains the engineering challenge of accounting for erroneous quantum operations on noisy hardware [3]. Compared to classical bits, quantum bits (qubits) are much more fragile and error-prone. The theory of Quantum Error Correction (QEC) [9, 15, 30, 38, 39] enables fault tolerant computation [7, 15, 37] using redundant qubits, but full fault tolerance is still prohibitively expensive for modern noisy devices—some 10^3 to 10^4 physical qubits are required to encode a single logical qubit [13, 25].

To reconcile quantum computation with NISQ computers, quantum compilers perform transformations for error mitigation [54] and noise-adaptive optimization [29]. To evaluate these compiler transformations, we must compare the error bounds of the source and compiled quantum programs.

Analyzing the error of quantum programs, however, is practically challenging. Although one can naively calculate the "distance" (i.e., error) between ideal and noisy outputs using their matrix representations [30], this approach is impractical for real-world quantum programs, whose matrix representations can be exponentially large—for example, a 20-qubit quantum circuit is represented by a $2^{20} \times 2^{20}$ matrix—too large to feasibly compute.

Rather than directly calculating the output error using matrix representations, an alternative approach employs *error metrics*, which can be computed more efficiently. A common error metric for quantum programs is the unconstrained diamond norm [1]. However, this metric only gives a *worst-case* error analysis: it is calculated only using quantum gates' noise models, and does not take into account any information about quantum state. In extreme cases, it overestimates errors by up to two orders of magnitude [55]. A more realistic metric must take input quantum state into account, since this also affects the output error.

The logic of quantum robustness (LQR) [21] was the first technique to use quantum state in the error metrics to compute tighter error bounds. This work introduces the (Q, λ)-diamond norm, which analyzes output errors given that the input quantum state satisfies some quantum predicate Q to degree λ. LQR extends the Quantum Hoare Logic [60] with the (Q, λ)-diamond norm to produce logical judgments of the form $(Q, \lambda) \vdash \widetilde{P} \leq \epsilon$, which deduces the error bound ϵ for a noisy program $\widetilde{P}$. While theoretically promising, this work raises open questions in practice. Consider the following sequence rule in their logic:

$$\frac{(Q_1, \lambda) \vdash \widetilde{P_1} \leq \epsilon_1 \qquad (Q_2, \lambda) \vdash \widetilde{P_2} \leq \epsilon_2 \qquad \{Q_1\} P_1 \{Q_2\}}{(Q_1, \lambda) \vdash (\widetilde{P_1}; \widetilde{P_2}) \leq \epsilon_1 + \epsilon_2}$$

It is unclear how to obtain a quantum predicate Q_2 that is both a valid postcondition after executing $\widetilde{P_1}$ while being strong enough to produce useful error bounds for $\widetilde{P_2}$.

This paper presents Gleipnir, an adaptive error analysis methodology for quantum programs that addresses the above challenges and answers the following three open questions: (1) How to compute suitable constraints for the input quantum state used by the error metrics? (2) How to reason about error bounds without manually verifying quantum programs with respect to pre- and postconditions? (3) How practical is computing verified error bounds for quantum programs, to evaluate the error mitigation performance of quantum compiler transformations?

First, in prior work, seaching for a non-trivial postcondition (Q, λ) for a given quantum program is prohibitively costly: existing methods either compute postconditions by

PL'18, January 01–03, 2018, New York, NY, USA
2018.

fully simulating quantum programs using matrix representations [60], or reduce this problem to an SDP (Semi-Definite Programming) problem whose size is exponential to the number of qubits used in the quantum program [61]. In practice, for large quantum programs ($\geq$ 20 qubits), these methods cannot produce any postconditions other than $(I, 0)$ (i.e., the identity matrix I to degree 0, analogous to a "true" predicate), reducing the (Q, λ)-diamond norm to the unconstrained diamond norm and failing to yield non-trivial error bounds.

To overcome this limitation, Gleipnir introduces the $(\hat{\rho}, \delta)$-diamond norm, a new error metric for input quantum states whose distance from some *approximated* quantum state $\hat{\rho}$ is bounded by δ. Given a quantum program and a predicate $(\hat{\rho}, \delta)$, Gleipnir computes its diamond norm by reducing it to a constant size SDP problem.

To obtain the predicate $(\hat{\rho}, \delta)$, Gleipnir uses Matrix Product States (MPS), a class of tensor networks, to represent and approximate quantum states. Rather than fully simulating the quantum program or producing an exponentially complex SDP problem, our MPS-based tensor network $TN(\rho_0, P)$ *approximates* $(\hat{\rho}, \delta)$ for some input state ρ_0 and program P, taking polynomial time with respect to the size of the MPS tensor network, the number of qubits, and the number of quantum gates. In contrast with prior work, our MPS-based approach is *adaptive*—one may adjust the approximation precision by varying the size of the MPS such that tighter error bounds can be computed using greater computational resources. Gleipnir provides more flexibility between the tight but inefficient full simulation and the efficient but unrealistic worst-case analysis.

Second, instead of verifying a predicate using Quantum Hoare Logic, Gleipnir develops a lightweight logic based on $(\hat{\rho}, \delta)$-diamond norms for reasoning about quantum program error, using judgments of the form:

$$(\hat{\rho}, \delta) \vdash \widetilde{P}_\omega \leq \epsilon$$

This judgement states that the error of the noisy program $\widetilde{P}_\omega$ under the noise model ω is upper-bounded by ϵ when the input state is constrained by $(\hat{\rho}, \delta)$. As shown in the sequence rule of our quantum error logic:

$$\frac{(\hat{\rho}, \delta) \vdash \widetilde{P}_{1\omega} \leq \epsilon_1 \quad TN(\hat{\rho}, P_1) = (\hat{\rho}', \delta') \quad (\hat{\rho}', \delta + \delta') \vdash \widetilde{P}_{2\omega} \leq \epsilon_2}{(\hat{\rho}, \delta) \vdash \widetilde{P}_{1\omega}; \widetilde{P}_{2\omega} \leq \epsilon_1 + \epsilon_2}$$

the approximated state $\hat{\rho}'$ and its δ' are computed using the MPS-based tensor network TN.

By computing $(\hat{\rho}, \delta)$, our sequence rule eliminates the cost of validating non-trivial postconditions. We prove the correctness of TN, which ensures that the resulting state of executing P_1 satisfies the predicate $(\hat{\rho}', \delta + \delta')$.

Third, we enable the practical error analysis of quantum programs and transformations, which was previously only theoretically possible but infeasible due to the limitations of prior work. To understand the scalability and limitation of our error analysis methodology, we conducted case studies using two classes of quantum programs that are expected to be most useful in the near-term—the Quantum Approximate Optimization Algorithm [12] and the Ising model [41]—with qubits ranging from 10 to 100. Our measurements show that, with 128-wide MPS networks, Gleipnir can always generate error bounds within 6 minutes. For small programs ($\leq$ 10 qubits), Gleipnir's error bounds are as precise as the ones generated using full simulation. For large programs ($\geq$ 20 qubits), Gleipnir's error bounds are 15% to 30% tighter than those calculated using unconstrained diamond norms, while full simulation invariably times out after 24 hours.

We explored Gleipnir's effectiveness in evaluating the error mitigation performance of quantum compiler transformations. We conducted a case study evaluating qubit mapping protocols [29], and showed that the performance ranking for different transformations using the error bounds generated by our methodology is consistent with the ranking using errors measured from the real-world experimental data.

Throughout this paper, we address the key practical limitations of error analysis for quantum programs. In summary, our main contributions are:

- The $(\hat{\rho}, \delta)$-diamond norm, a new error metric constrained by input quantum state, that can be efficiently computed using constant-size SDPs.
- An MPS-based tensor network approach to adaptively compute the quatum predicate $(\hat{\rho}, \delta)$.
- A lightweight logic for reasoning about quantum error bounds without the need to verify quantum predicates.
- Case studies using quantum programs and transformations on real quantum devices, demonstrating the feasability of adaptive quantum error analysis for computing verified error bounds for quantum programs and evaluating the error mitigation perforamce of quantum compilation.

2 Quantum Programming Background

Notation. In this paper, we use Dirac notation, or "bra-ket" notation, to represent quantum states. The "ket" notation $|\psi\rangle$ denotes a column vector, which corresponds to a pure quantum state; the "bra" notation $\langle\psi|$ denotes its conjugate transpose, a row vector. $\langle\phi|\psi\rangle$ represents the inner product of two vectors, and $|\psi\rangle\langle\phi|$ the outer product. We use ρ to denote a density matrix, a matrix that represents mixed quantum state. U usually denotes a unitary matrix which represents quantum gates while $U^\dagger$ denotes its conjugate transpose; curly letters like $\mathcal{U}$ denote noisy or ideal quantum operations, represented by maps between density matrices (superoperators); upper case Greek letters such as Φ represent quantum noise as superoperators.

2.1 Quantum computing basics

Quantum states. The simplest quantum state is a quantum bit—a *qubit*. Unlike a classical bit, a qubit's state can

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},\ H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 1. Matrix representations of common quantum gates. X denotes a bit flip, Z denotes a phase flip, H denotes a Hadamard gate, and $CNOT$ denotes a controlled NOT gate.

be the superposition of two logical states, $|0\rangle$ and $|1\rangle$, that correspond to classical logical states 0 and 1. In general, a qubit is a unit vector in the 2-dimensional Hilbert space $\mathbb{C}^2$, with $|0\rangle := [1, 0]^\dagger$ and $|1\rangle := [0, 1]^\dagger$. In Dirac's notation, we represent a qubit as $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$.

Generally speaking, the state of a quantum program may comprise many qubits. An n-qubit state can be represented by a unit vector in 2^n-dimensional Hilbert space $\mathbb{C}^{2^n}$. For example, a 3-qubit state can be described by an 8-dimensional complex vector, which captures a superposition of 8 basis states, $|000\rangle$, $|001\rangle$, $|010\rangle$, ..., $|111\rangle$.

Besides the pure quantum states described above, there are also *classically mixed* quantum states, i.e., noisy states. An n-qubit mixed state can be represented by a $2^n \times 2^n$ *density matrix* $\rho = \sum_i p_i\,|\phi_i\rangle\,\langle\phi_i|$, which states that the state has p_i probability to be $|\phi_i\rangle$. For example, a mixed state with half probability of $|0\rangle$ and $|1\rangle$ can be represented by $\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} = I/2$, where I is the identity matrix.

Quantum gates. Quantum states are manipulated by the application of *quantum gates*, described by unitary matrix representations [30]. Figure 1 shows the matrix representations of some common gates. Applying an operator U to a quantum state $|\phi\rangle$ results in the state $U\,|\phi\rangle$, and applying it to a density matrix $\rho = \sum_i p_i\,|\phi_i\rangle\,\langle\phi_i|$ gives $U\rho U^\dagger$. For example, the bit flip gate X maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$, while the Hadamard gate H maps $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$. There are also multi-qubit gates, such as $CNOT$, which does not change $|00\rangle$ and $|01\rangle$ but maps $|10\rangle$ and $|11\rangle$ to each other. Applying a gate on a subset of qubits will not change other qubits. For example, applying the X gate to the first qubit of $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ will result in $\frac{|10\rangle+|01\rangle}{\sqrt{2}}$. This can be seen as an extension $X \otimes I$ of the matrix to a larger space using a tensor product.

Quantum measurements. Measurements extract classical information from quantum states, and collapse the quantum state according to *projection matrices* M_0 and M_1. When we measure some state ρ, we will obtain the result 0 with collapsed state $M_0\rho M_0^\dagger/p_0$ and probability $p_0 = \mathrm{tr}(M_0\rho M_0^\dagger)$, or the result 1 with collapsed state $M_1\rho M_1^\dagger/p_1$ and probability $p_1 = \mathrm{tr}(M_1\rho M_1^\dagger)$. Both quantum gates and quantum measurements act linearly on density matrices, and can be expressed as *superoperators*, completely positive trace-preserving maps $\mathcal{E} \in L(\mathcal{H}) : \mathcal{H}_n \to \mathcal{H}_m$ where $\mathcal{H}_n$ is the density matrix space of dimension n and L is the space of linear operators.

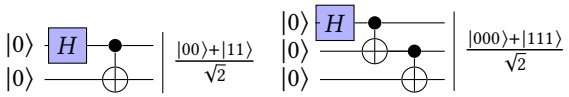


Figure 2. Two quantum circuits, producing the 2-qubit (left) and 3-qubit (right) GHZ states.

2.2 Quantum programs

Quantum programs comprise a configuration of quantum gates and measurements, called a quantum circuit. Graphically, qubits are represented as wires, and gates as boxes joining the wires; CNOT gates are represented by a dot on the control qubit linked with an $\bigoplus$ on the other qubit.

Example 2.1 (GHZ circuit)**.** The Greenberger–Horne–Zeilinger (GHZ) state [16] is a class of important entangled quantum states, used in many quantum communication protocols [19]. The simplest GHZ state is the 2-qubit GHZ state, which is $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ in Dirac notation. Figure 2 shows a typical graphical representation of a quantum circuit that produces the 2-qubit GHZ state.

Syntax. The syntax of quantum programs is as follows:

$$P ::= \texttt{skip} \mid P_1; P_2 \mid U(q_1, \ldots, q_k)$$
$$\mid \texttt{if}\ q = |0\rangle\ \texttt{then}\ P_0\ \texttt{else}\ P_1$$

Each component behaves similarly to its classical counterpart: `skip` denotes the empty program; $P_1; P_2$ sequences programs; $U(q_1, \ldots, q_k)$ applies the k-qubit gate U to the qubits $q_1, \ldots, q_k$; `if` $q = |0\rangle$ `then` P_0 `else` P_1 measures the qubit q, executes P_0 if the result is 0, and executes P_1 otherwise. The only difference between classical and quantum programs is that the measurement in the `if` statement will collapse the state, and the branch is executed on the collapsed state. Using this syntax, the 2-qubit GHZ state circuit in Fig. 2 is written as:

$$H(q_0); CNOT(q_0, q_1)$$

Note that this work does not currently consider advanced quantum program constructs such as quantum loops, as these are not likely to be supported on near-term quantum machines.

Denotational semantics. The denotationl semantics of quantum programs are defined as superoperators acting on density matrices ρ, and are shown in Fig. 3. An empty program keeps the state unchanged; a sequence of operations are applied to the state one by one; a single quantum gate is directly applied as a superoperator; a measurement branch statement maps the state into a classical mix of the two results from executing the two branches.

2.3 Quantum errors

Quantum programs are always noisy, and that noise may (undesirably) perturb the quantum state. For example, the

$$[[\texttt{skip}]](\rho) := \rho$$
$$[[P_1; P_2]](\rho) := [[P_2]]([[P_1]](\rho))$$
$$[[U(q_1, \ldots, q_k)]](\rho) := U\rho U^\dagger$$
$$[[\texttt{if } q = |0\rangle \texttt{ then } P_0 \texttt{ else } P_1]](\rho) := [[P_0]](M_0 \rho M_0^\dagger) + [[P_1]](M_1 \rho M_1^\dagger)$$

Figure 3. Denotational semantics of quantum programs.

bit flip noise flips the state of a qubit with probability p. This noise can be represented by a superoperator Φ such that:

$$\Phi(\rho) = (1-p)\rho + pX\rho X$$

i.e., the state remains the same with probability $1-p$ and changes to $X\rho X$ with probability p, where X is the matrix representation of the bit flip gate (Fig. 1). Generally, all effects from quantum noise can be represented by superoperators.

Noisy quantum programs. The noise model ω specifies the noisy version $\widetilde{U}_\omega$ of each gate U on the targeting noisy device and can then specify noisy quantum programs $\widetilde{P}_\omega$. The noisy semantics $[[P]]_\omega$ of program P can be defined as the semantics $[[\widetilde{P}_\omega]]$ of the noisy program $\widetilde{P}_\omega$, whose semantics are similar to that of a noiseless program. The rules of skip, sequence, and measurement statements remain the same, while for gate application, the noisy version of each gate is applied as follows:

$$[[U(q_1, \ldots, q_k)]]_\omega(\rho) = [[\widetilde{U}_\omega(q_1, \ldots, q_k)]](\rho) = \widetilde{\mathcal{U}}_\omega(\rho)$$

where $\widetilde{\mathcal{U}}_\omega$ is the superoperator representation of $\widetilde{U}_\omega$.

Metrics for quantum errors. To quantitatively evaluate the effect of noise, we need to measure some notion of "distance" between quantum states. The *trace distance* T measures the distance between the noisy state ρ_n and the ideal, noiseless state ρ_{Id}:

$$\mathrm{T}(\rho_n, \rho_{\text{Id}}) := \frac{1}{2}\|\rho_n - \rho_{\text{Id}}\|_1 = \frac{1}{2}\max_P \; P(\rho_n - \rho_{\text{Id}})$$

where P is a positive semidefinite matrix with trace 1, and $||\cdot||_p$ is the Schatten-p norm, defined as:

$$||\cdot||_p := \left(\mathrm{tr}(\cdot^\dagger \cdot)^{\frac{p}{2}}\right)^{\frac{1}{p}}$$

The trace distance is the maximum statistical distance over all possible measurements of two quantum states. Note that trace distance cannot be directly calculated without information about the entire two quantum states.

The *diamond norm* metric is typically used to obtain a *worst case* error bound. The diamond norm between two superoperators $\mathcal{U}$ and $\mathcal{E}$ is defined as:

$$||\mathcal{U} - \mathcal{E}||_\diamond := \frac{1}{2}\max_{\rho:\ \mathrm{tr}(\rho)=1} \mathrm{T}\Big(\mathcal{U} \otimes \mathcal{I}_A(\rho),\ \mathcal{E} \otimes \mathcal{I}_A(\rho)\Big) = \max_{\rho:\ \mathrm{tr}(\rho)=1} ||\mathcal{U} \otimes \mathcal{I}_A(\rho) - \mathcal{E} \otimes \mathcal{I}_A(\rho)||_1$$

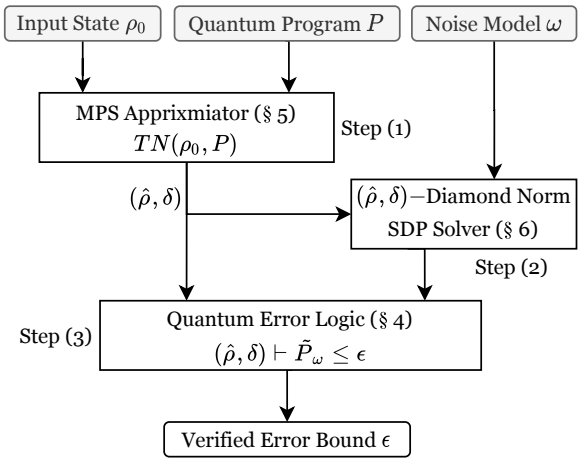


Figure 4. Gleipnir workflow.

Intuitively, this formula calculates the maximum trace distance between the output state after applying the erroneous operation versus applying the noiseless operation, for any *arbitrary* input state. Diamond norms can be efficiently computed by simple Semi-Definite Programs (SDP)[57].

However, as shown by the Wallman-Flammia bound [55], diamond norms may overestimate errors by up to two orders of magnitude, precluding its application in more precise analyses of noisy quantum programs. The diamond norm metric fails to incorporate information about the quantum state of the circuit that may help tighten the error bound. For example, a bit flip error (X gate) does nothing to the state $\frac{\sqrt{2}}{2}(|0\rangle + |1\rangle)$ (the state is unchanged after flipping $|0\rangle$ and $|1\rangle$), but flips the $|1\rangle$ state to $|0\rangle$. However, both trace distance and diamond norm are agnostic to the input state, and thus limit our ability to tightly bound the errors of quantum circuits.

(Q, λ)−diamond norm [21] is a more fine-grained metric:

$$||\mathcal{U} - \mathcal{E}||_\diamond := \max_{\rho:\ \mathrm{tr}(\rho)=1, \mathrm{tr}(Q\rho)\geq\lambda} ||\mathcal{U} \otimes \mathcal{I}_A(\rho) - \mathcal{E} \otimes \mathcal{I}_A(\rho)||_1$$

Unlike the unconstrained diamond norm, the (Q, λ)−diamond norm constrains the input state to satisfy the predicate Q to degree λ; specifically, the input state ρ must satisfy $\mathrm{tr}(Q\rho) \geq \lambda$. The (Q, λ)−diamond norm may produce tighter error bounds than the unconstrained diamond norm by utilizing quantum state information, but leaves open the problem of practically computing a non-trivial predicate Q.

3 Gleipnir Workflow

To use the input quantum state to tighten the computed error bound, Gleipnir introduces a new constrained diamond norm, $(\hat{\rho}, \delta)$-diamond norm, and a judgment $(\hat{\rho}, \delta) \vdash \widetilde{P}_\omega \leq \epsilon$ to reason about the error of quantum circuits. Gleipnir uses Matrix Product State (MPS) tensor networks to approximate quantum state and compute the predicate $(\hat{\rho}, \delta)$.

$$\frac{}{(\hat{\rho},\delta)\vdash \widetilde{P}_{\omega}\le 0}\ \textsc{Skip} \qquad \frac{\|\widetilde{\mathcal{U}}_{\omega}-\mathcal{U}\|_{(\hat{\rho},\delta)}\le\epsilon}{(\hat{\rho},\delta)\vdash \widetilde{U}_{\omega}(q_1,\ldots)\ \le\epsilon}\ \textsc{Gate} \qquad \frac{(\hat{\rho},\delta)\vdash \widetilde{P}_{1\omega}\le\epsilon_1 \quad TN(\hat{\rho},P_1)=(\hat{\rho}',\delta') \quad (\hat{\rho}',\delta+\delta')\vdash \widetilde{P}_{2\omega}\le\epsilon_2}{(\hat{\rho},\delta)\vdash \widetilde{P}_{1\omega};\widetilde{P}_{2\omega}\ \le\epsilon_1+\epsilon_2}\ \textsc{Seq}$$

$$\frac{(\hat{\rho},\delta')\vdash \widetilde{P}_{\omega}\le\epsilon' \qquad \epsilon'\le\epsilon \qquad \delta'\ge\delta}{(\hat{\rho},\delta)\vdash \widetilde{P}_{\omega}\le\epsilon}\ \textsc{Weaken} \qquad \frac{(\hat{\rho}_0,\delta)\vdash \widetilde{P}_{0\omega}\le\epsilon \qquad (\hat{\rho}_1,\delta)\vdash \widetilde{P}_{1\omega}\le\epsilon}{(\hat{\rho},\delta)\vdash \left(\texttt{if}\ \ q=|0\rangle\ \texttt{then}\ \ \widetilde{P}_{0\omega}\ \ \texttt{else}\ \ \widetilde{P}_{1\omega}\right)\ \le(1-\delta)\epsilon+\delta}\ \textsc{Meas}$$

Figure 5. Inference rules of the quantum error logic.

Figure 4 illustrates Gleipnir's workflow for reasoning about the error bound of some quantum program P with input state ρ_0 and noise model ω of quantum gates on the target device:

1. Gleipnir first approximates the quantum state $\hat{\rho}$ and its distance δ to the ideal state ρ using MPS tensor networks $TN(\rho_0,P)=(\hat{\rho},\delta)$ (see §5).
2. Gleipnir then uses the constrained $(\hat{\rho},\delta)$-diamond norm metric to bound errors of noisy quantum gates given a noise model ω of the target device. Gleipnir converts the problem of efficiently computing the $(\hat{\rho},\delta)$-diamond norm to solving a polynomial-size SDP problem, given $(\hat{\rho},\delta)$ computed in step 1 (see §6).
3. Gleipnir employs a lightweight quantum error logic to compute the error bound of $\widetilde{P}_{\omega}$ using the predicate $(\hat{\rho},\delta)$ computed in step 1 and the error bounds for all used quantum gates generated by the SDP solver in step 2 (see §4).

Throughout this paper, we will return to the GHZ state circuit (Example 2.1) as our running example. We will use the program $H(q_0);CNOT(q_0,q_1)$, the input state $|00\rangle\langle 00|$, and the noise model ω, describing the noisy gates $\widetilde{H}_{\omega}$ and $\widetilde{CNOT}_{\omega}$. Following the steps described above, we will use Gleipnir to obtain the final judgment of:

$$(|00\rangle\langle 00|,\ 0)\vdash \left(\widetilde{H}_{\omega}(q_0)\ ;\ \widetilde{CNOT}_{\omega}(q_0,q_1)\right)\le\epsilon$$

where ϵ is the total error bound of the noisy program.

4 Quantum Error Logic

We first introduce our lightweight logic for reasoning about quantum program error bounds. In this section, we treat MPS tensor networks and the algorithm to compute the$(\hat{\rho},\delta)$-diamond norm as black boxes, deferring their discussion to later sections (§5 and §6, respectively).

The $(\hat{\rho},\delta)$-diamond norm is defined as follows:

$$||\mathcal{U}-\mathcal{E}||_{(\hat{\rho},\delta)}:=\frac{1}{2}\max_{\substack{\rho:\ \mathrm{tr}(\rho)=1,\\ T(\rho,\hat{\rho})\le\delta}}\mathrm{T}\Big(\mathcal{U}\otimes\mathcal{I}_A(\rho),\ \mathcal{E}\otimes\mathcal{I}_A(\rho)\Big)$$

That is, a diamond norm with the additional constraint that the ideal input density matrix of ρ needs to be within distance δ of $\hat{\rho}$, i.e., $T(\rho,\hat{\rho})\le\delta$.

We use the judgment $(\hat{\rho},\delta)\vdash\widetilde{P}_{\omega}\le\epsilon$ to convey that when running the noisy program $\widetilde{P}_{\omega}$ on an input state whose trace distance is at most δ from $\hat{\rho}$, the trace distance between the noisy and noiseless outputs of program P is at most ϵ under the noise model ω of the underline device.

The five inference rules of our quantum error logic are shown in Fig. 5. The Skip rule states that an empty program does not produce any noise. The Gate rule states that we can bound the error of a gate step by calculating the gate's $(\hat{\rho},\delta)$-diamond norm under the noise model ω. The Weaken rule states that the same error bound holds when we strengthen the precondition with a smaller approximation bound δ'. The Seq rule states that the errors of a sequence can be summed together with the help of the tensor network approximator TN. The Meas rule bounds the error in an `if` statement, with δ probability that the result of measuring the noisy input differs from measuring state ρ, causing the wrong branch to executed. Otherwise, the probability that the correct branch is executed is $1-\delta$. We multiply this probability by the error incurred by the correct branch, and add it to the probability of taking the incorrect branch, to obtain the error incurred by executing a quantum conditional statement.

Our error logic contains two external components: (1) $TN(\rho,P)=(\hat{\rho},\delta)$ uses the tensor network approximator to approximate $[\![P]\!](\rho)$, obtaining $\hat{\rho}$ and an approximation error bound δ; and (2) $\|\cdot\|_{(\hat{\rho},\delta)}$ is the $(\hat{\rho},\delta)$-diamond norm, which characterizes the error bound generated by a single gate under the noise model ω. The algorithms used to compute these components are explained in §5 and §6, respectively, while the soundness proof of our inference rules is given in supplementary material A.

We demonstrate how these rules can be applied to the GHZ circuit of Example 2.1 as follows. The input state is $\rho=|00\rangle\langle 00|$, and the program is $\widetilde{H}_{\omega}(q_0);\widetilde{CNOT}_{\omega}(q_0,q_1)$. We first compute the constrained diamond norm $\epsilon_1=\|\widetilde{\mathcal{H}}_{\omega}-\mathcal{H}\|_{(\rho,0)}$, and apply the Gate rule to obtain:

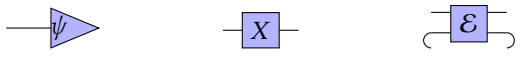
$$(\rho,0)\vdash\widetilde{H}_{\omega}(q_0)\le\epsilon_1$$

Then, we use the tensor network approximator to compute $TN(\rho,H(q_0))$, whose result is $(\hat{\rho},\delta)$. Using such a predicate, we compute the $(\hat{\rho},\delta)$-diamond norm $\epsilon_2=\|\widetilde{\mathcal{CNOT}}_{\omega}-\mathcal{CNOT}\|_{(\hat{\rho},\delta)}$. Applying the Gate rule again, we obtain:

$$(\hat{\rho},\delta)\vdash\widetilde{CNOT}_{\omega}(q_0,q_1)\le\epsilon_2$$

Finally, we apply the Seq rule:

$$(\rho,0)\vdash\left(\widetilde{H}_{\omega}(q_0);\widetilde{CNOT}_{\omega}(q_0,q_1)\right)\le\epsilon_1+\epsilon_2$$



(a) Vector (rank 1) **(b)** Matrix (rank 2) **(c)** Superoperator (rank 4)

Figure 6. Tensor network representation of various tensors.

which gives the error bound of the noisy program, $\epsilon_1 + \epsilon_2$.

5 Quantum State Approximation

Gleipnir uses tensor networks to adaptively compute the contraints of input quantum state using an approximate state $\hat{\rho}$ and its distance δ from the ideal state ρ. We provide the background of tensor networks in §5.1, present how to use tensor networks to approximate quantum states in §5.2, and then give an example in §5.3.

5.1 Tensor network

Tensors. Tensors describe the the multilinear relationship between sets of objects in vector spaces, and can be represented by multi-dimensional arrays. The *rank* of a tensor indicates the dimensionality of its array representation: vectors have rank 1, matrices rank 2, and superoperators rank 4 (since they operate over rank 2 density matrices).

Contraction. Tensor contraction generalizes vector inner products and matrix multiplication. A contraction between two tensors specifies an index for each tensor, sums over these indices and produces a new tensor. The contraction of two tensors with ranks a and b will have rank $a + b - 2$; for example, if we contract the first index in 3-tensor A and the second index in 2-tensor B, the output will be a 3-tensor:

$$(A \times_{1,2} B)_{jkl} = \sum_t A_{tjk} B_{lt}$$

Tensor product. The tensor product is calculated like an outer product; if two tensors have ranks a and b respectively, their tensor product is a rank $a + b$ tensor. For example, the tensor product of 2-tensor A and 2-tensor B is a 4-tensor:

$$(A \otimes B)_{ijkl} = A_{ij} B_{kl}$$

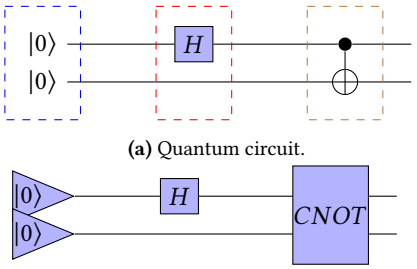
Tensor networks. Tensor network (TN) representation is a graphical calculus for reasoning about tensors, with an intuitive representation of various quantum objects. Introduced in the 1970s by Penrose [33], this notation is used in quantum information theory [11, 43, 51–53, 59], as well as in other such fields as machine learning [10, 44, 45].

As depicted in Fig. 6, tensor networks consist of nodes and edges. [1] Each node represents a tensor, and each edge out of the node represents an index of the tensor. As illustrated in Fig. 7, the resulting network will itself constitute a whole tensor, with each open-ended edge representing one index for the final tensor. The graphical representation of a quantum program can be directly interpreted as a tensor network. For example, the 2-qubit GHZ state circuit in Fig. 2 can be represented by a tensor network, as shown in Fig. 8.

[1] Note that the shape of the nodes does not have any mathematical meaning; it is merely used to distinguish different types of tensors.

(a) Matrix multiplication AB **(b)** Outer product $|\psi\rangle\langle\psi|$

Figure 7. Tensor network representation for two matrix operations. In general, tensor contractions are represented by linking edges, and tensor products by juxtaposition.



(a) Quantum circuit.

(b) Tensor network.

Figure 8. The GHZ state, represented as a quantum circuit (a) and a tensor network (b). When we evaluate the output of the circuit, we can see that the input state $|00\rangle$ (enclosed in the blue box), the H gate $H \otimes I$ (enclosed in the middle red box), and the $CNOT$ gate (enclosed in the brown box. When evaluating the tensor network in (b), the output is the same as the program output, $(|00\rangle + |11\rangle)/\sqrt{2}$.

Transforming tensor networks. To speed up the evaluation of a large tensor network, we can apply reduction rules to transform and simplify the network structure. In Table 1, we summarize some common reduction rules we use. The Gate Contraction rule transforms a vector ψ and a matrix U connected to it into a new vector ϕ that is the product of U and ψ. The Superoperator Application rule transforms a superoperator $\mathcal{E}$ and a matrix ρ connected to it into a matrix $\hat{\rho}$ that represents the application of the superoperator $\mathcal{E}$ to ρ. The Singular Value Decomposition (SVD) rule transforms a matrix M into the product of three matrices: U, Σ, and $V^\dagger$. The special matrix $\Sigma = \sum_j \sigma_j |j\rangle\langle j|$ is a diagonal matrix, often represented by a diamond in graph. By dropping small singular values in the diagonal matrix Σ, we obtain a simpler tensor network which closely approximates the original one.

5.2 Approximate quantum states

In this section, we describe our tensor network approximator algorithm computing $TN(\rho, P) = (\hat{\rho}, \delta)$, such that the trace distance between our approximation $\hat{\rho}$ and the perfect output $[\![P]\!](\rho)$ satisfies $T\big(\hat{\rho}, [\![P]\!](\rho)\big) \leq \delta$. At each stage of the algorithm, we use Matrix Product State (MPS) tensor networks, a special class of tensor networks, to approximate quantum states. MPS uses $2n$ matrices to represent a 2^n-length vector, which greatly reduces the computational cost. The MPS

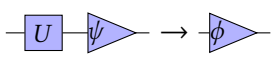
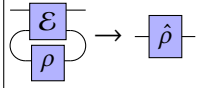
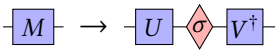
	Gate Contraction	Superoperator Application	Singular Value Decomposition
Tensor network			
Dirac notation	$U\|\psi\rangle = \|\phi\rangle$	$\mathcal{E}(\rho) = \hat{\rho}$	$M = \sum_j \sigma_j U\|j\rangle\langle j\|V^\dagger$
Rank	1	2	2

Table 1. Examples of tensor network transformations for basic quantum operations.

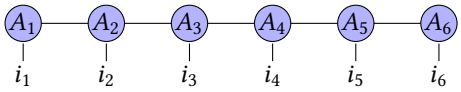


Figure 9. An example of matrix product state of six qubits.

tensor networks take a size w as an argument, which determines the space of representable states. When w is not big enough to represent all possible quantum states, the MPS is just an *approximation* to quantum states, whose approximation bound depends on w. The MPS representation with size w of a quantum state ψ (represented as a vector) is:

$$|\psi\rangle_{\text{MPS}} \coloneqq \sum_{i_1,\ldots,i_n} A_1^{(i_1)} A_2^{(i_2)} \cdots A_n^{(i_n)} |i_1 i_2 \cdots i_n\rangle$$

where $A_1^{(i_1)}$ is a row vector of dimension w, $A_2^{(i_2)}, \ldots, A_{n-1}^{(i_{n-1})}$ are $w \times w$ matrices, and $A_n^{(i_n)}$ is a column vector of dimension w. i_j is the value of a basis $|i_1 i_2 \cdots i_n\rangle$ at position j, which can be 0 or 1. For example, to represent the 3-qubit state $\big(|000\rangle + |010\rangle + |001\rangle\big)/3$ in MPS, we should find matrices $A_1^{(0)}, A_1^{(1)}, A_2^{(0)}, A_2^{(1)}, A_3^{(0)}, A_3^{(1)}$ such that

$$A_1^{(0)} A_2^{(0)} A_3^{(0)} = A_1^{(0)} A_2^{(1)} A_3^{(0)} = A_1^{(0)} A_2^{(0)} A_3^{(1)} = \frac{1}{3},$$

while $A_1^{(i_1)} A_2^{(i_2)} A_3^{(i_3)} = 0$ for all $(i_1, i_2, i_3) \neq (0,0,0), (0,1,0)$, or $(0,0,1)$.

$A_i^{(0)}$ and $A_i^{(1)}$ can be seen together as a 3-tensor A_i (A_0 and A_n are 2-tensors) where the superscript is taken as the third index besides the two indices of the matrix. The MPS in total can be seen as a tensor network in Fig. 9. $A_1, \ldots, A_n$ are linked together in a line, while $i_1, \ldots, i_n$ are open wires.

Our approximation algorithm works by initializing the MPS to the input state in vector form, and applying each gate from the quantum program to the MPS, approximating the intermediate state at each step as an MPS and computing the distance between MPS and the ideal state. Since MPS only needs to maintain $2n$ tensors, i.e., $A_1^{(0)}, A_1^{(1)}, A_2^{(0)}, A_2^{(1)}, \cdots, A_n^{(0)}, A_n^{(1)}$, this procedure can be done efficiently with a polynomial time complexity. After applying all quantum gates, we obtain an MPS that approximates the output state of the quantum program, as well as an approximation bound by summing together all accumulated approximation errors incurred by the approximation process. Our approximation algorithm consists of the following stages:

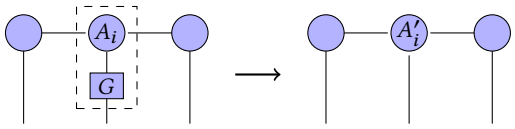


Figure 10. Applying a one-qubit gate to an MPS. We contract the MPS node for the qubit and the gate (in the dashed box), resulting in another 3-tensor MPS node.

Initialization. Let $|s_1 s_2 \cdots s_n\rangle$ be the input state for an n-qubit quantum circuit. For all $k \in [1, n]$, we initialize $A_k^{(s_k)} = E$ and $A_k^{(1-s_k)} = 0$, where E is the matrix that $E_{1,1} = 1$ and $E_{i,j} = 0$ for all $i \neq 1, j \neq 1$.

Applying one-qubit gates. Applying a one-qubit gate on an MPS always results in an MPS, and thus does not incur any approximation error. For a single-qubit gate G on qubit i, we update the tensor A_i to A_i' as follows:

$$A_i'^{(s)} = \sum_{s' \in \{0,1\}} G_{ss'} A_i^{(s')} \qquad \text{for } s = 0 \text{ or } 1$$

In the tensor network representation, such application amounts to contracting the tensor for the gate with A_i (see Fig. 10).

Applying two-qubit gates. If we are applying a two-qubit gate G on two adjacent qubits i and $i+1$, we only need to modify A_i and A_{i+1}. We first contract A_i, A_{i+1} to get an $2w \times 2w$ matrix M:

$$\begin{bmatrix} A_i^{(0)} \\ A_i^{(1)} \end{bmatrix} \begin{bmatrix} A_{i+1}^{(0)} & A_{i+1}^{(1)} \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} = M$$

Then, we apply the two-qubit gate to it.

$$M'_{ij} = \sum_{k,l} G_{ijkl} M_{kl}$$

We then need to decompose this new matrix M' back into two tensors. We first apply the Singular Value Decomposition rule on the contracted matrix:

$$M' = U \Sigma V^\dagger$$

When w is not big enough to represent all possible quantum states, M' introduces approximation errors and may not be a contraction of two tensors. Thus, we *truncate* the lower half of the singular values in Σ, enabling the tensor

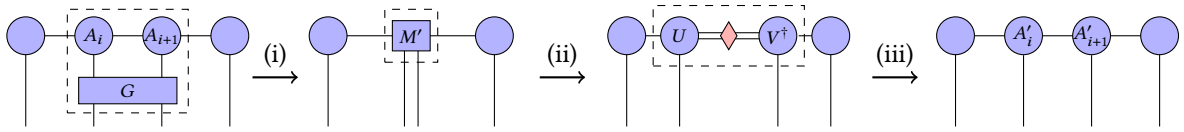


Figure 11. Applying a two-qubit gate on two adjacent qubits to the MPS, via (i) node contraction, (ii) singular value decomposition, and (iii) singular value truncation with re-normalization.

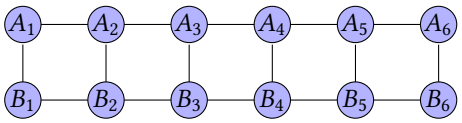


Figure 12. Tensor network representation of the inner product of two MPSs. An open wire of one MPS is linked with an open wire of another, which denotes the summation over $i_1, \ldots, i_n$.

decomposition while reducing the error:

$$\Sigma \approx \begin{bmatrix} \Sigma' & 0 \\ 0 & 0 \end{bmatrix}$$

Therefore, we arrive at a new MPS whose new tensors A'_i and A'_{i+1} are calculated as follows:

$$\begin{bmatrix} A_i^{(0)\prime} & * \\ A_i^{(1)\prime} & * \end{bmatrix} = U, \begin{bmatrix} A_{i+1}^{(0)\prime} & A_{i+1}^{(1)\prime} \\ * & * \end{bmatrix} = \Sigma' V$$

where $*$ is part of that we drop. After truncation, we renormalize the state to a norm-1 vector.

Figure 11 shows the above procedure in tensor network form by (1) first applying Gate Contracting rule for A_i, A_{i+1} and G, (2) using Singular Value Decomposition rule to decompose the contracted tensor, (3) truncating the internal edge to w width, and finally (4) calculating the updated A'_i and A'_{i+1}. If we want to apply a two-qubit gate to non-adjacent qubits, we add swap gates to move the two qubits together, and apply the gate on the two adjacent qubits.

Bounding approximation errors. When applying 2-qubit gates, we compute an MPS to approximate the gate application. Each time we do so, we must estimate the error due to this approximation. Since the truncated values themselves comprise an MPS state, we may determine the error by simply calculating the trace distance between the states before and after truncation.

The trace distance of two MPS states can be calculated from the inner product of these two MPS:

$$\delta := \mathrm{T}\big(|\phi\rangle\langle\phi|, |\psi\rangle\langle\psi| \big) = \sqrt{1 - |\langle\phi|\psi\rangle|^2}.$$

The inner product of two states $|\psi\rangle$ and $|\phi\rangle$ (represented using A and B in their MPS forms) is defined as follows:

$$\langle\psi|\phi\rangle = \sum_{i_1,\ldots,i_n} \left\langle A_1^{(i_1)} \cdots A_n^{(i_n)}, B_1^{(i_1)} \cdots B_n^{(i_n)} \right\rangle$$

Figure 12 shows its tensor network graphical representation.

In our approximation algorithm, we can iteratively calculate the distance from qubit 1 to qubit n by first determining:

$$D_1 = A_1^{(0)} B_1^{(0)\dagger} + A_1^{(1)} B_1^{(1)\dagger}$$

Then, we repeatedly apply tensors to the rest of qubits:

$$D_i = A_i^{(0)} D_{i-1} B_i^{(0)\dagger} + A_i^{(1)} D_{i-1} B_i^{(1)\dagger}$$

leading us to the final result of $D_n = \langle\psi|\phi\rangle$. In the tensor network graphical representation, this algorithm is a left-to-right contraction, as shown in Fig. 13.

Given the calculated distance of each step, we must combine them to obtain the overall approximation error. For some arbitrary quantum program with t 2-qubit gates, let the truncation errors be $\delta_1, \delta_2, \ldots, \delta_t$ when applying the 2-qubit gates $g_1, g_2, ..., g_t$. The final approximation error is $\delta = \sum_{i=1}^{t} \delta_i$. To show this, we consider the approximation of one 2-qubit gate. Let $|\psi\rangle$ denote some quantum state, and $|\hat{psi}\rangle$ its approximation with bounded error δ_0. After applying a 2-qubit gate G to the approximate MPS state, we obtain the truncated result $|\phi\rangle$ with bounded error δ_1. We now have:

$$\begin{aligned} \| G|\psi\rangle - |\phi\rangle \| &\le \| G|\psi\rangle - G|\hat{\psi}\rangle \| + \| G|\hat{\psi}\rangle - |\phi\rangle \| \\ &= \| |\psi\rangle - |\hat{\psi}\rangle \| + \| G|\hat{\psi}\rangle - |\phi\rangle \| \\ &= \delta_0 + \delta_1. \end{aligned} \quad (1)$$

where $\| |\psi\rangle - |\phi\rangle \| = T(|\psi\rangle\langle\psi|, |\phi\rangle\langle\phi|)$. The inequality holds because of the triangular inequality of quantum state distance, and the fact that G is unitary, thus preserving the norm. Repeating this for each step, we know that the total approximation error is bounded by the sum of all approximation errors. The local density operator also has an approximation error, which is also bound by the sum because partial traces do not increase trace distance.

Complexity analysis. The time complexity of all the operations above scales polynomially with respect to the MPS size w, number of qubits n, and number of gates m in the program. To be precise, applying a one-qubit gate requires only matrix addition, with $O(w^2)$ time. Applying a two-qubit gate requires matrix multiplication and SVD, in $O(w^3)$ time. Computing inner product of two MPS (e.g. for contraction) requires $O(n)$ of matrix multiplications, incurring an overall time complexity of $O(nw^3)$. Since the algorithm scanning all m gates in the program, the total complexity is $O(mnw^3)$.

Although a *perfect* approximation (i.e., a full simulation) requires an MPS size that scales exponentially with respect to the number of qubits, our approximation algorithm allows Gleipnir to be configured with smaller MPS sizes, sacrificing

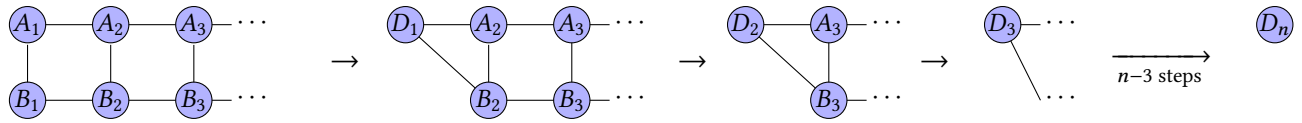


Figure 13. Contraction of the inner product of two MPS. We first contract A_1 and B_1 to get D_1. Then contract D_1, A_2 and B_2 to D_2. And then D_2, A_3 and B_3 to D_3. Repeating this process will result a single tensor node D_n, i.e., the final answer.

some precision in favor of efficiency and enabling its practical use for real-world quantum programs.

Correctness. From the quantum program semantics defined in Fig. 3, we know that we can compute the output state by applying all gates in the program in sequence. Following Eq. (1), we know the total error bound for our approximation algorithm is bounded by sum of the bounds of each step. Thus, we can conclude that our algorithm correctly approximates the output state, and correctly bounds the approximation error in doing so:

Theorem 5.1. *Let the output of our approximation algorithm be* $(\hat{\rho}, \delta) = \mathrm{TN}(\rho, P)$. *The trace distance between the approximation and perfect output is bound by* δ*:*

$$\mathrm{T}\Big(\hat{\rho},\ [\![P]\!](\rho)\Big) \leq \delta.$$

5.3 Example: GHZ circuit

We revisit the GHZ circuit in Fig. 2 to walk through how we approximate quantum states with tensor networks. This same technique can be applied to larger and more complex quantum circuits, discussed in §7.

Approximation using 2-wide MPS. Since the program only contains two qubits, an MPS with size $w = 2$ can already perfectly represent all possible quantum states such that no approximation error will be introduced. Assume the input state is $|00\rangle$. First, we initialize all the tensors based on the input state $|00\rangle$:

$$A_1^{(0)} = [1, 0],\ A_1^{(1)} = [0, 0],\ A_2^{(0)} = [1, 0]^T,\ A_2^{(1)} = [0, 0]^T$$

Then, we apply the first H gate to qubit 1, changing only $A_1^{(0)}$ and $A_1^{(1)}$:

$$A_1^{(0)} = [1, 0]/\sqrt{2}, \qquad A_1^{(1)} = [1, 0]/\sqrt{2}$$

To apply the CNOT gate on qubit 1 and 2, we first compute matrix M and M':

$$M = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \end{bmatrix}, \quad M' = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

We then decompose M' using SVD to get the new MPS:

$$A_1^{(0)} = [1, 0], \qquad A_1^{(1)} = [0, 1],$$
$$A_2^{(0)} = [1/\sqrt{2}, 0]^T, \qquad A_2^{(1)} = [0, 1/\sqrt{2}]^T$$

We can see that the output will be $\hat{\rho} = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ and $\delta = 0$, since $A_1^{(0)}A_2^{(0)} = A_1^{(1)}A_2^{(1)} = 1/\sqrt{2}$ while other values of i_0, i_1 result 0.

Approximation using 1-wide MPS. To show how we calculate the approximation error, we use the simplest form of MPS with size $w = 1$. All $A_i^{(j)}$ will become numbers.

We first initialize the MPS to represent $|00\rangle$:

$$A_1^{(0)} = 1,\ A_1^{(1)} = 0,\ A_2^{(0)} = 1,\ A_2^{(1)} = 0.$$

Then, we apply the H gate to qubit 1:

$$A_1^{(0)} = 1/\sqrt{2},\ A_1^{(1)} = 1/\sqrt{2},\ A_2^{(0)} = 1,\ A_2^{(1)} = 0.$$

After that, we apply the CNOT gate. We compute M and M':

$$M = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \end{bmatrix}, \quad M' = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

We decompose M' using SVD:

$$U = V^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}.$$

Since there are 2 non-zero singular values, we need to drop the lower half. Finally, we obtain A_1' and A_2':

$$A_1^{(0)} = 1,\ A_1^{(1)} = 0,\ A_2^{(0)} = 1/\sqrt{2},\ A_2^{(1)} = 0.$$

We renormalize the MPS:

$$A_1^{(0)} = 1,\ A_1^{(1)} = 0,\ A_2^{(0)} = 1,\ A_2^{(1)} = 0.$$

Thus, the output approximate state is $|00\rangle$.

To calculate the approximation error bound, we represent the part we drop as an MPS B:

$$B_1^{(0)} = 0,\ B_1^{(1)} = 1,\ B_2^{(0)} = 0,\ B_2^{(1)} = \sqrt{2}.$$

Let the unnormalized final state be $|A\rangle$, the dropped state be $|B\rangle$. Then, the final output is $\sqrt{2}\,|A\rangle$ and the ideal output is $|A\rangle + |B\rangle$. The trace distance between the state is

$$\delta = \sqrt{1 - |\,\langle\sqrt{2}A|A+B\rangle^2\,|} = 1/\sqrt{2}.$$

Therefore, the final output is $\hat{\rho} = |00\rangle\langle 00|$ and $\delta = 1/\sqrt{2}$.

6 Computing the $(\hat{\rho}, \delta)$-Diamond Norm

In §4, we introduced our quantum error logic using the $(\hat{\rho}, \delta)$-diamond norm, while treating its computation algorithm as a black box. In this section, we describe how to efficiently calculate the $(\hat{\rho}, \delta)$-diamond norm given $(\hat{\rho}, \delta, \mathcal{U}, \mathcal{E})$. We show that $(\hat{\rho}, \delta)$ can be transformed into (Q, λ) in supplementary material B.

Constrained diamond norm. In $(\hat{\rho}, \delta)$-diamond norm, the input state ρ_{in} is constrained by

$$T(\hat{\rho}, \rho_{\text{in}}) \leq \delta$$

We first compute the local density matrix of $\hat{\rho}$ which is ρ', and since trace distance does not increase, we have $T(\rho', \rho) \leq \delta$. Recall for any matrix ρ, we have $\|\rho\|_F \leq T(\rho)$, where $\|\cdot\|_F$ is the Frobenius norm which is the square root of the sum of all elements in a matrix. Therefore, from $T(\rho', \rho) \leq \delta$, we know that $\|\rho' - \rho\|_{\text{F}} < \delta$, which means that $\text{tr}(\rho'\rho) \geq \|\rho'\|_F(\|\rho'\|_F - \delta)$. Then, to compute the $(\hat{\rho}, \delta)$-diamond norm, we extend the result of Watrous [57] by adding the constraint of $\text{tr}(\rho'\rho) \geq \|\rho'\|_F(\|\rho'\|_F - \delta)$, such that $(\hat{\rho}, \delta)$-diamond norm can be computed by the following SDP:

Theorem 6.1. *The $(\hat{\rho}, \delta)$-diamond norm $||\Phi||_{(\hat{\rho},\delta)}$ can be solved by SDP in Eq. (2).*

$$\begin{array}{ll} \text{maximize} & \text{tr}(J(\Phi)W) \\ \text{subject to} & I \otimes \rho \succcurlyeq W \\ & \text{tr}(\rho'\rho) \geq \|\rho'\|_F(\|\rho'\|_F - \delta) \\ & W \succcurlyeq 0,\ \rho \succcurlyeq 0,\ \text{tr}(\rho) = 1 \end{array} \tag{2}$$

where J is the Choi-Jamiolkowski isomorphism [8] and $\Phi = \mathcal{U} - \mathcal{E}$. Let the optimal value of SDP in Eq. (2) be ϵ. We conclude that:

$$\|\Phi\|_{(\hat{\rho},\delta)} \leq \epsilon$$

SDP size. The size of SDP in Eq. (2) is exponential with respect to the number of qubits of any *quantum gate*, rather than of the whole program. Since near-term (NISQ) quantum computers are unlikely to support quantum gates with greater than 2 qubits, we can treat the size of the SDP problem as a constant, for the purposes of discussing its time complexity.

Computing local density matrix. The local density matrix represents the local information of a quantum state. It is defined using a partial trace on the (global) density for the part of the state we want to observe. For example, the local density operator on the first qubit of $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is $[[0.5, 0.5], [0.5, 0.5]]$, meaning that the first qubit of the state is half $|0\rangle$ and half $|1\rangle$.

In Eq. (2), we need to compute the local density matrix ρ' of $\hat{\rho}$ about the qubit(s) Q that the noise represented by Φ acts on. $\hat{\rho}$ is represented by an MPS. The calculation of a local density operator of a MPS works similarly to how we calculate inner products, except the wire i_k where k is a qubit that we want to observe.

7 Evaluation

In this section, we evaluate Gleipnir on using a set of realistic near-term quantum programs. We compare the bounds given by Gleipnir to the bounds given by other methods, as well as the error we experimentally measured from a IBM's real quantum device. All simulations and our approximations are performed on an Intel Xeon W-2175 (28 cores @ 4.3 GHz) 62 GB memory, and a 512 GB Intel SSD Pro 600p.

7.1 Simulation

We evaluated Gleipnir on several important quantum programs, under a sample noise model containing the most common type of quantum noises. We compared the bound produced by Gleipnir with the worse-case bound given by the unconstrained diamond norm.

Noise model. In our experiments, our quantum circuits are configured such that, with probability $p = 10^{-4}$, each noisy one-qubit gate has either a bit flip (X):

$$\Phi(\rho) = (1-p)\rho + pX\rho X$$

or a phase flip (Z):

$$\Phi(\rho) = (1-p)\rho + pZ\rho Z$$

Each two qubit gate also has a bit flip or phase flip on its first qubit.

Framework configuration. For the approximator, we can adjust the size of the MPS network, depending on available computational resources; the larger the size, the tighter error bound. In all experiments, we use an MPS of size 128.

Baseline. To evaluate the performance of the error bound given by Gleipnir, we compared it with a worst-case bound calculated using the unconstrained diamond norm (see §2.3). For each noisy quantum gate, we first compute its unconstrained diamond norm distance to the perfect gate, and obtain the worst-case bound by summing all unconstrained diamond norms. The unconstrained diamond norm distance of a bit-flipped gate and a perfect gate is given by:

$$\begin{aligned} \|\Phi - I\|_\diamond &= \|(pX \circ X + (1-p)I) - I\|_\diamond \\ &= p\|X \circ X - I\|_\diamond \\ &= p \end{aligned}$$

where $X \circ X$ denotes the function that maps ρ to $X\rho X$. The diamond norm of a phase-flipped gate is derived similarly. Therefore, the total noise is bounded by np, where n is the number of noisy gates, due to additivity of diamond norms. Because every gate has a noise, the worst case bound is simply proportional to the number of gates in the program.

Benchmark	Qubit number	Gate number	Gleipnir bound ($\times 10^{-4}$)	Running time (s)	LQR [21] with full simulator ($\times 10^{-4}$)	Running time (s)	Worst-case bound ($\times 10^{-4}$)
QAOA_line_10	10	27	0.05	2.77	0.05	215.2	27
Isingmodel10	10	480	335.6	31.6	335.6	4701.8	480
QAOARandom20	20	160	136.6	19.8	-	(timed out)	160
QAOA4reg_20	20	160	138.8	12.5	-	(timed out)	160
QAOA4reg_30	30	240	207.0	25.8	-	(timed out)	240
Isingmodel45	45	2265	1739.4	338.0	-	(timed out)	2265
QAOA50	50	399	344.1	58.7	-	(timed out)	399
QAOA75	75	597	517.2	113.7	-	(timed out)	597
QAOA100	100	677	576.7	191.9	-	(timed out)	677

Table 2. Simulation results of our model ($w = 128$) and the baseline on different quantum programs, showing the bounds given by Gleipnir's $(\hat{\rho}, \delta)$-diamond norm, the (Q, λ)-diamond norm with full simulation, and the unconstrained diamond norm. Simulations time out if they run for longer than 24 hours. Note that the worst case bound is directly proportional to the number of gates.

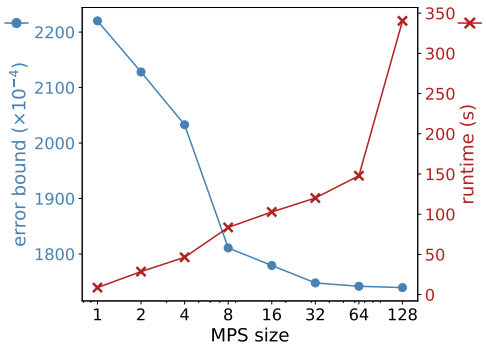


Figure 14. Simulation results of our model on program `Isingmodel45` using different MPS size.

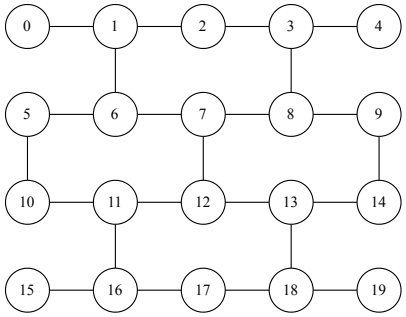


Figure 15. The coupling map of the IBM Boeblingen quantum computer, where each node represents a qubit. Only qubit pairs with a connecting edge can be used to implement a 2-qubit gate.

We also compared our error bound with what we obtain from LQR [21], using a full quantum program simulator to generate best quantum predicate. This approach's running time is exponential to the number of qubits, and times out (runs for longer than 24 hours) on programs with ≥ 20 qubits.

Programs. We analyzed two classes of quantum programs that are expected to be most useful in the near-term, namely:

- The *Quantum Approximate Optimization Algorithm (QAOA)* [12], which can be used to solve combinatorial optimization problems. We use it to find the max-cut for various graphs, with qubit sizes from 10 to 100.
- The *Ising model* [41], which is a thermodynamic model for magnets widely used in quantum mechanics. We run the Ising model with sizes 10 and 45.

Evaluation. Results are shown in Table 2. Gleipnir's bounds are 15% $\sim$ 30% tighter than what the unconstrained diamond norm gives, on large quantum circuits with qubit sizes ≥ 20. On small qubit size circuits, our bound is as strong as the exponential time method with full simulation.

We also evaluated how MPS size impacts the performance of Gleipnir. As we can see for the `Isingmodel45` program (Fig. 14), larger MPS sizes result in tighter error bounds, at the cost of longer run times, with marginal returns beyond a certain size. We found that a size of 128 seemed to perform best for our candidate programs, though in general, this parameter can be adjusted according to precision requirements and the availability of computational resources. Note that one cannot feasibly compute the precise error bound of the `Isingmodel45` program, since that requires computing the $2^{45} \times 2^{45}$ matrix representation of the program's output.

7.2 Evaluating quantum compilation error mitigation

To demonstrate that Gleipnir can be used to evaluate the error mitigation performance of quantum compilers for real quantum computers today, we designed a small experiment based on the noise-adaptive qubit mapping problem [5, 29]. When executing a quantum program on a real quantum computer, a quantum compiler must decide which physical qubit that each logical qubit should be mapped to, in accordance with the quantum computer's coupling map (e.g., Fig. 15). Since quantum devices do not have uniform noise across qubits, a quantum compiler's mapping protocol should aim to map qubits such that the quantum program is executed with as little noise as possible.

Mapping	Gleipnir bound	Measured error
0-1-2	0.211	0.160
1-2-3	0.128	0.073
2-3-4	0.162	0.092

Table 3. Error bounds generated by Gleipnir on different mappings compared with the noise we observed experimentally.

Experiment design. We compared three different qubit mappings of the 3-qubit GHZ circuit (see Fig. 2), $q_0 - q_1 - q_2$, $q_1 - q_2 - q_3$, and $q_2 - q_3 - q_4$, where q_i represents ith physical qubit. We ran our circuit on our quantum computer with each qubit mapping, and measured the output to obtain a classical probability distribution. We computed the measured error by taking the statistical distance of this distribution from the distribution of the ideal output state $(|000\rangle + |111\rangle)/\sqrt{2}$. We also used Gleipnir to compute the noise bound for each mapping, based on our quantum computer's noise model. Because the trace distance represents the maximum possible statistical distance of any measurement on two quantum states (see §2.3), the statistical distance we computed should be bounded by the trace distance computed by Gleipnir.

Experiment setup. We conducted our experiment using the IBM Quantum Experience[22] platform, using the IBM Boeblingen 20-qubit device to run our quantum programs (Fig. 15). Because Gleipnir needs a noise model to compute its error bound, we constructed a model for the device using publicly available data from IBM [22] in addition to measurements from tests we ran on the device. We identified two different types of noise:

1. *Gate errors* occur because gate operations cannot be performed perfectly. We used quantum process tomography [31, 36] to test each individual gate (H and $CNOT$ in our case) and reconstruct their noisy superoperator representation.
2. *Qubit decoherence errors* occur because qubits are not perfectly isolated, and may interact with their environment. We obtained the device's decoherence errors from the IBMQ database's $T1$ and $T2$ device data [22].

Results. Our experimental results are shown in Table 3. Gleipnir's bounds are consistent with the real noise level, and successfully predicts the noise levels of different mapping: $1-2-3$ has the least noise, while $0-1-2$ has the most. This illustrates how Gleipnir can be used to inform the design of noise-adaptive mapping protocols.

8 Related Work

Error bounding quantum programs. Robust projective quantum Hoare logic [62] is an extension of Quantum Hoare Logic that supports error bounding using the worst-case diamond norm. In contrast, Gleipnir uses the more fine-grained $(\hat{\rho}, \delta)$–diamond norm to provide tighter error bounding.

Like Gleipnir, LQR [21] is a framework for formally reasoning about quantum program errors, using the (Q, λ)-diamond norm as its error metric. LQR supports reasoning about programs that use more advanced quantum computing features, such as quantum loops. However, it does not specify any practical method for obtaining non-trivial predicates. In contrast, Gleipnir can automatically compute $(\hat{\rho}, \delta)$ predicates using its TN algorithm. We further show these computed predicates can be reduced to (Q, λ) predicates (see supplementary material B). In other words, our quantum error logic can be understood as an implementation refining LQR: $(\hat{\rho}, \delta)$ predicates computed using Gleipnir can be used to obtain non-trivial postconditions for the quantum Hoare triples required by LQR's sequence rule, which, by the soundness of our TN algorithm, are guaranteed to be valid.

Error simulation. Current error simulation methods can be roughly divided into two classes: (1) direct simulation methods based on solving the Schrödinger's equation or the master equation [27], which do not scale beyond a few qubits [32]; and (2) approximate methods, based on either Clifford circuit approximation [6, 17, 18, 26] or classical sampling methods with Monte-Carlo simulations [28, 42, 48, 50]. These methods are efficient, but only work on specific classes of quantum circuits. In contrast, Gleipnir can be applied to general quantum circuits, and scales well beyond 20 qubits.

Resource estimation beyond error. Quantum compilers such as Qiskit Terra [2] and ScaffCC [23] perform entanglement analysis for quantum programs. The QuRE [47] toolbox provides coarse-grained resource estimation for fault-tolerant implementations of quantum algorithms. On the theoretical side, quantum resource theories also consider the estimation of coherence [46, 58], entanglement [34, 35], and magic state stability [20, 49, 56]. However, these frameworks are still based on the matrix representation of quantum states and are only applicable to very small quantum programs.

Tensor network approximation. Multi-dimensional tensor networks such as PEPS [24] and MERA [14] may model quantum states more precisely than MPS. However, they are computationally impractical: contracting higher-dimensional tensor networks involves tensors with orders greater than 4, which are prohibitively expensive to manipulate.

9 Conclusion

We have presented Gleipnir, a methodology for computing verified error bounds of quantum programs and evaluating the error mitigation performance of quantum compiler transformations. Our simulation results show that Gleipnir provides up to 33% tighter error bounds in quantum circuits with qubits ranging from 10 to 100 and the generated error bounds are consistent with the ones measured using real quantum devices.

References

[1] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. 1998. Quantum circuits with mixed states. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 20–30.

[2] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[3] C. G. Almudever, L. Lao, X. Fu, N. Khammassi, I. Ashraf, D. Iorga, S. Varsamopoulos, C. Eichler, A. Wallraff, L. Geck, A. Kruth, J. Knoch, H. Bluhm, and K. Bertels. 2017. The engineering challenges in quantum computing. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 836–845. https://doi.org/10.23919/DATE.2017.7927104

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510. https://doi.org/10.1038/s41586-019-1666-5

[5] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh. 2019. MUQUT: Multi-Constraint Quantum Circuit Mapping on NISQ Computers: Invited Paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–7.

[6] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. 2019. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum* 3 (Sept. 2019), 181. https://doi.org/10.22331/q-2019-09-02-181

[7] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. 2017. Roads towards fault-tolerant universal quantum computation. *Nature* 549, 7671 (2017), 172–179. https://doi.org/10.1038/nature23460

[8] Man-Duen Choi. 1975. Completely positive linear maps on complex matrices. *Linear Algebra Appl.* 10, 3 (1975), 285 – 290. https://doi.org/10.1016/0024-3795(75)90075-0

[9] Simon J Devitt, William J Munro, and Kae Nemoto. 2013. Quantum error correction for beginners. *Reports on Progress in Physics* 76, 7 (Jun 2013), 076001. https://doi.org/10.1088/0034-4885/76/7/076001

[10] Stavros Efthymiou, Jack Hidary, and Stefan Leichenauer. 2019. TensorNetwork for Machine Learning. *ArXiv* abs/1906.06329 (2019).

[11] G. Evenbly and G. Vidal. 2009. Algorithms for entanglement renormalization. *Physical Review B* 79, 14 (Apr 2009). https://doi.org/10.1103/physrevb.79.144108

[12] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[13] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. 2012. Surface codes: Towards practical large-scale quantum computation. *Physical Review A* 86, 3 (Sep 2012). https://doi.org/10.1103/physreva.86.032324

[14] Vittorio Giovannetti, Simone Montangero, and Rosario Fazio. 2008. Quantum multiscale entanglement renormalization ansatz channels. *Physical review letters* 101, 18 (2008), 180503.

[15] Daniel Gottesman. 2009. An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation. arXiv:0904.2557 [quant-ph]

[16] Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. 1989. Going beyond Bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*. Springer, 69–72.

[17] Mauricio Gutiérrez, Conor Smith, Livia Lulushi, Smitha Janardan, and Kenneth R. Brown. 2016. Errors and pseudothresholds for incoherent and coherent noise. *Phys. Rev. A* 94 (Oct 2016), 042338. Issue 4. https://doi.org/10.1103/PhysRevA.94.042338

[18] Mauricio Gutiérrez, Lukas Svec, Alexander Vargo, and Kenneth R. Brown. 2013. Approximation of realistic errors by Clifford channels and Pauli measurements. *Physical Review A* 87, 3 (Mar 2013). https://doi.org/10.1103/physreva.87.030302

[19] Mark Hillery, Vladimír Bužek, and André Berthiaume. 1999. Quantum secret sharing. *Physical Review A* 59, 3 (1999), 1829.

[20] Mark Howard and Earl Campbell. 2017. Application of a Resource Theory for Magic States to Fault-Tolerant Quantum Computing. *Physical Review Letters* 118, 9 (Mar 2017). https://doi.org/10.1103/physrevlett.118.090501

[21] Shih-Han Hung, Kesha Hietala, Shaopeng Zhu, Mingsheng Ying, Michael Hicks, and Xiaodi Wu. 2019. Quantitative Robustness Analysis of Quantum Programs. *Proc. ACM Program. Lang.* 3, POPL, Article 31 (Jan. 2019), 29 pages. https://doi.org/10.1145/3290344

[22] IBMQ 2016. IBM-Q Experience. https://www.research.ibm.com/ibm-q/

[23] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. 2015. ScaffCC: Scalable compilation and analysis of quantum programs. *Parallel Comput.* 45 (2015), 2–17.

[24] Jacob Jordan, Roman Orús, Guifre Vidal, Frank Verstraete, and J Ignacio Cirac. 2008. Classical simulation of infinite-size quantum lattice systems in two spatial dimensions. *Physical review letters* 101, 25 (2008), 250602.

[25] E. Knill. 2005. Quantum computing with realistically noisy devices. *Nature* 434, 7029 (Mar 2005), 39–44. https://doi.org/10.1038/nature03350

[26] Easwar Magesan, Daniel Puzzuoli, Christopher E. Granade, and David G. Cory. 2013. Modeling quantum noise for efficient testing of fault-tolerant circuits. *Physical Review A* 87, 1 (Jan 2013). https://doi.org/10.1103/physreva.87.012324

[27] Nancy Makri and Dmitrii E. Makarov. 1995. Tensor propagator for iterative quantum time evolution of reduced density matrices. I. Theory. *The Journal of Chemical Physics* 102, 11 (2019/11/13 1995), 4600–4610. https://doi.org/10.1063/1.469508

[28] A. Mari and J. Eisert. 2012. Positive Wigner Functions Render Classical Simulation of Quantum Computation Efficient. *Physical Review Letters* 109, 23 (Dec 2012). https://doi.org/10.1103/physrevlett.109.230503

[29] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi. 2019. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. arXiv:1901.11054 [quant-ph]

[30] Michael A. Nielsen and Isaac L. Chuang. 2011. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th ed.). Cambridge University Press, New York, NY, USA.

[31] J. L. O'Brien, G. J. Pryde, A. Gilchrist, D. F. V. James, N. K. Langford, T. C. Ralph, and A. G. White. 2004. Quantum Process Tomography of a Controlled-NOT Gate. *Phys. Rev. Lett.* 93 (Aug 2004), 080502. Issue 8. https://doi.org/10.1103/PhysRevLett.93.080502

[32] Hakop Pashayan, Stephen D. Bartlett, and David W. Gross. 2017. From estimation of quantum probabilities to simulation of quantum circuits.

[33] Roger Penrose. 1971. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications* 1 (1971), 221–244.

[34] M. Piani, M. Horodecki, P. Horodecki, and R. Horodecki. 2006. Properties of quantum nonsignaling boxes. *Phys. Rev. A* 74 (Jul 2006), 012305. Issue 1. https://doi.org/10.1103/PhysRevA.74.012305

[35] Martin B. Plenio and S. Virmani. 2005. An introduction to entanglement measures. arXiv:quant-ph/0504163 [quant-ph]

[36] J. F. Poyatos, J. I. Cirac, and P. Zoller. 1997. Complete Characterization of a Quantum Process: The Two-Bit Quantum Gate. *Phys. Rev. Lett.* 78 (Jan 1997), 390–393. Issue 2. https://doi.org/10.1103/PhysRevLett.78.390

[37] John Preskill. 1997. Fault-tolerant quantum computation. arXiv:quant-ph/9712048 [quant-ph]

[38] John Preskill. 1998. Lecture notes for physics 229: Quantum information and computation. (1998).

[39] John Preskill. 1998. Reliable quantum computers. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454, 1969 (Jan 1998), 385–410. https://doi.org/10.1098/rspa.1998.0167

[40] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *ArXiv e-prints* (jan 2018). https://doi.org/10.22331/q-2018-08-06-79 arXiv:1801.00862

[41] Google AI Quantum et al. 2020. Hartree-Fock on a superconducting qubit quantum computer. *Science* 369, 6507 (2020), 1084–1089.

[42] Robert Raußendorf, Juani Bermejo-Vega, E. Tyhurst, Cihan Okay, and Michael Zurel. 2019. Phase space simulation method for quantum computation with magic states on qubits.

[43] U. Schollwöck. 2005. The density-matrix renormalization group. *Reviews of Modern Physics* 77, 1 (Apr 2005), 259–315. https://doi.org/10.1103/revmodphys.77.259

[44] Edwin Stoudenmire and David J Schwab. 2016. Supervised Learning with Tensor Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4799–4807. http://papers.nips.cc/paper/6211-supervised-learning-with-tensor-networks.pdf

[45] Edwin Stoudenmire and David J Schwab. 2016. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*. 4799–4807.

[46] Alexander Streltsov, Gerardo Adesso, and Martin B. Plenio. 2017. Colloquium: quantum coherence as a resource.

[47] Martin Suchara, John Kubiatowicz, Arvin I. Faruque, Frederic T. Chong, Ching-Yi Lai, and Gerardo Paz. 2013. QuRE: The Quantum Resource Estimator toolbox. In *2013 IEEE 31st International Conference on Computer Design, ICCD 2013, Asheville, NC, USA, October 6-9, 2013*. 419–426. https://doi.org/10.1109/ICCD.2013.6657074

[48] Victor Veitch, Christopher Ferrie, David W. Gross, and Joseph Emerson. 2012. Negative quasi-probability as a resource for quantum computation.

[49] Victor Veitch, S A Hamed Mousavian, Daniel Gottesman, and Joseph Emerson. 2014. The resource theory of stabilizer quantum computation. *New Journal of Physics* 16, 1 (jan 2014), 013009. https://doi.org/10.1088/1367-2630/16/1/013009

[50] Victor Veitch, Nathan Wiebe, Christopher Ferrie, and Joseph Emerson. 2013. Efficient simulation scheme for a class of quantum optics experiments with non-negative Wigner representation.

[51] F. Verstraete, V. Murg, and J.I. Cirac. 2008. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics* 57, 2 (Mar 2008), 143–224. https://doi.org/10.1080/14789940801912366

[52] Guifré Vidal. 2003. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Physical Review Letters* 91, 14 (Oct 2003). https://doi.org/10.1103/physrevlett.91.147902

[53] G. Vidal. 2008. Class of Quantum Many-Body States That Can Be Efficiently Simulated. *Physical Review Letters* 101, 11 (Sep 2008). https://doi.org/10.1103/physrevlett.101.110501

[54] Joel J. Wallman and Joseph Emerson. 2016. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A* 94, 5 (Nov 2016). https://doi.org/10.1103/physreva.94.052325

[55] Joel J Wallman and Steven T Flammia. 2014. Randomized benchmarking with confidence. *New Journal of Physics* 16, 10 (Oct 2014), 103032. https://doi.org/10.1088/1367-2630/16/10/103032

[56] Xin Wang, Mark M Wilde, and Yuan Su. 2019. Quantifying the magic of quantum channels. *arXiv preprint arXiv:1903.04483* (2019).

[57] John Watrous. 2012. Simpler semidefinite programs for completely bounded norms. arXiv:1207.5726 [quant-ph]

[58] Andreas J. Winter and Dong Yuan Yang. 2016. Operational Resource Theory of Coherence. *Physical review letters* 116 12 (2016), 120404.

[59] Christopher J. Wood, Jacob D. Biamonte, and David G. Cory. 2011. Tensor networks and graphical calculus for open quantum systems. arXiv:1111.6950 [quant-ph]

[60] Mingsheng Ying. 2016. *Foundations of Quantum Programming*. Morgan Kaufmann.

[61] Mingsheng Ying, Shenggang Ying, and Xiaodi Wu. 2017. Invariants of quantum programs: characterisations and generation. *ACM SIGPLAN Notices* 52, 1 (2017), 818–832.

[62] Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. ACM, New York, NY, USA, 1149–1162. https://doi.org/10.1145/3314221.3314584