Fast estimation of outcome probabilities for quantum circuits

Hakop Pashayan,[1,2] Oliver Reardon-Smith,[3] Kamil Korzekwa,[3] and Stephen D. Bartlett[4]

[1] *Institute for Quantum Computing and Department of Combinatorics and Optimization, University of Waterloo, ON, N2L 3G1 Canada*
[2] *Perimeter Institute for Theoretical Physics, Waterloo, ON, N2L 2Y5 Canada*
[3] *Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, 30-348 Kraków, Poland*
[4] *Centre for Engineered Quantum Systems, School of Physics, The University of Sydney, Sydney, NSW 2006, Australia*

(Dated: February 1, 2021)

We present two classical algorithms for the simulation of universal quantum circuits on n qubits constructed from c instances of Clifford gates and t arbitrary-angle Z-rotation gates such as T gates. Our algorithms complement each other by performing best in different parameter regimes. The ESTIMATE algorithm produces an additive precision estimate of the Born rule probability of a chosen measurement outcome with the only source of run-time inefficiency being a linear dependence on the stabilizer extent (which scales like $\approx 1.17^t$ for T gates). Our algorithm is state-of-the-art for this task: as an example, in approximately 25 hours (on a standard desktop computer), we estimated the Born rule probability to within an additive error of 0.03, for a 50-qubit, 60 non-Clifford gate quantum circuit with more than 2000 Clifford gates. The COMPUTE algorithm calculates the probability of a chosen measurement outcome to machine precision with run-time $O\left(2^{t-r}(t-r)t\right)$ where r is an efficiently computable, circuit-specific quantity. With high probability, r is very close to $\min\{t, n-w\}$ for random circuits with many Clifford gates, where w is the number of measured qubits. COMPUTE can be effective in surprisingly challenging parameter regimes, e.g., we can randomly sample Clifford+T circuits with $n = 55$, $w = 5$, $c = 10^5$ and $t = 80$ T-gates, and then compute the Born rule probability with a run-time consistently less than 10^4 seconds using a single core of a standard desktop computer. We provide a C+Python implementation of our algorithms.

I. INTRODUCTION

With the rapid advancement in experimental control over noisy intermediate-scale quantum (NISQ) systems [1], claims of quantum advantage [2] have recently been made using several different platforms [3, 4]. Along with the enormous challenges in building complex quantum devices that can exhibit quantum advantage, a complementary but equally challenging problem is how to test if these devices are operating as intended. From the current NISQ era until we achieve universal fault-tolerant quantum computers, we will need tools to compare classically-computed theoretical predictions with the observed frequency of particular events generated by a quantum device. Techniques such as direct fidelity estimation [5] already rely on such comparisons. Google's recent demonstration [3] also used classical algorithms for predicting features of the expected distribution of outcomes.

As the exact calculation of Born rule probabilities becomes increasingly difficult for larger and larger quantum circuits, Born rule probability estimation techniques are becoming increasingly important. Innovations in techniques for Born rule probability estimation will impact on the broader field of quantum characterization, verification and validation [6]. Classical techniques for Born rule probability estimation also have an increasingly important role in evaluating and developing proposals for NISQ device applications. For example, proposals for new kernel-method-based quantum machine learning algorithms can be better evaluated using classical algorithms for additive polynomial precision estimation of Born rule probabilities [7].

Brute force simulation algorithms such as Schrödinger-style [8], Feynman-style [9–11] or hybrid simulators [12] offer high precision general purpose classical simulation capabilities for universal quantum circuits. However, such simulations can be extremely resource intensive for moderate circuit width (number of qubits $n \approx 40$) and/or depth. Alternatively, there exist efficiently classically simulable families of (non-universal) quantum circuits [13–17]. In particular, the Gottesman-Knill theorem makes it possible to classically simulate thousands of qubits with hundreds of thousands of gates provided that we restrict to so-called stabilizer circuits [13].

In contrast to these two extremes, Aaronson and Gottesman [16] were the first to present a classical simulation algorithm that is efficient for stabilizer circuits but can also simulate non-stabilizer circuits with a run-time cost that is exponential in the number of non-stabilizer gates (non-Clifford gates). A limitation of this work is that the run-time does not depend on the specifics of the additional non-stabilizer gates. Thus, their simulator pays a heavy run-time penalty for introducing a small number of non-stabilizer gates even if these are only marginally far from stabilizer gates. Research to overcome this limitation falls into two broad categories: Born rule probability estimators based on using a quasi-probabilistic representation of the density matrix [18–25], and pure-state sampling simulators [26–29]. While quasi-probabilistic simulators produce additive polynomial precision estimates of Born rule probabilities,

pure state sampling simulators have better run-time scaling in the exponential component. (Specifically, the latter's performance scales linearly rather than quadratically in a quantity known as the *stabilizer extent* [30], which is a measure of how far a pure state is from the nearest stabilizer state.) However, pure state sampling simulators such as that of Refs. [22, 29, 30] output samples from the approximate quantum outcome distribution rather than estimates of Born probabilities. Used as a black-box, $O\left(\epsilon^{-2}\right)$ samples from such a sampling algorithm can be used to produce an additive ϵ-error Born probability estimator, but this contributes a further factor of $O\left(\epsilon^{-2}\right)$ to the run-time that in many practical regimes is already heavily dominated by the polynomial scaling components.

In this paper, we present an additive polynomial precision estimator of Born rule probabilities. Our estimator is actually a pair of distinct algorithms that work as part of a larger procedure, utilizing their respective performance advantages in complementary regimes, and is state-of-the-art for the task. The first of our pair of algorithms — the ESTIMATE algorithm — uses the pure state formalism, ensuring that our simulator scales linearly in the stabilizer extent, quickly outperforming quasi-probabilistic simulators as the number of non-stabilizer elements increases. Additionally, by extending existing methods and developing a number of new techniques, our estimation algorithm performs many orders of magnitude faster than those of Refs. [29, 30] in certain practically relevant parameter regimes.

The second algorithm of our pair is COMPUTE, a classical algorithm that computes exact Born rule probabilities (up to machine precision). The run-time of this algorithm depends exponentially on the effective number $(t - r)$ of non-Clifford gates, where t is the original number of non-Clifford gates and r is a circuit-specific parameter which can be efficiently pre-computed. This parameter r can generally be as large as the minimum of t and the number of unmeasured qubits $(n - w)$. Our COMPUTE algorithm is complementary to ESTIMATE, performing particularly well for large circuits consisting of many Clifford elements, as observed through testing on random circuits. In this setting, we observe that r is generically large ($r \approx \min\{t, n - w\}$). Alternatively, when r is small, our ESTIMATE algorithm outperforms COMPUTE due to its run-time dependence on r^3. This should be contrasted with the corresponding factor of t^3 in the run-time of the sampling algorithm of Ref. [29] and the factor of n^3 in the run-time of the 'sum over Cliffords' algorithm of Ref. [30].

The paper is structured as follows. In Section II, we provide a brief review of Born rule probability estimation and previous results, and a high-level overview of our estimator, including how it works, the run-time of its various components, and its performance for various parameter regimes of quantum circuits. In Sec. III we give details of the ancillary COMPRESS algorithm that directly leads to our COMPUTE algorithm, and is used as a pre-processing to the ESTIMATE algorithm. The primary contribution of this paper, the ESTIMATE algorithm, is presented and analyzed in Secs. IV and V, with Sec. IV devoted to the crucial RAWESTIM subroutine of ESTIMATE. We conclude with an outlook in Sec. VI. We also provide several appendices containing further details on aspects of the new algorithms and methods we present.

II. OVERVIEW

In this section, we present a high-level summary of the key results of our paper, including a statement of the problem of Born rule probability estimation and related research on this topic, as well as our main contributions of a suite of algorithms to perform Born rule probability estimation of universal quantum circuit families.

A. Statement of problem

Quantum circuits that initiate in a computational basis state, evolve under Clifford unitary transformations and are measured in the computational basis are classically simulable by the Gottesman-Knill theorem [13, 16]. The group of Clifford unitary transformations is generated by the gate-set $\{S, H, CX\}$ (and contains CZ):

$$S \equiv \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad CX \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad CZ \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{1}$$

This gateset is promoted to universality by the inclusion of a non-Clifford gate, which is a diagonal gate

$$T_\phi \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}, \tag{2}$$

where $\phi \in (0, \pi/2)$ can be arbitrary. A standard choice is the T gate [31], defined with $\phi = \pi/4$.

We consider a system composed of n qubits initially prepared in the state $|0\rangle^{\otimes n}$. The system then evolves according to a unitary transformation U to the final state $U|0\rangle^{\otimes n}$. We consider circuits where U is constructed via a sequence of elementary gates from the gate-set consisting of S, H, CX, CZ and T_{ϕ_j} gates, where each $\phi_j \in (0, \pi/2)$ can be arbitrary. We will call such a description of the circuit an *elementary description* of U and reserve the variables c, h and t to respectively denote the number of all Clifford gates, Hadamard gates and non-Clifford gates occurring in this description. Associated with the non-Clifford gates T_{ϕ_j}, for $j \in [t]$, are the non-stabilizer single qubit states $|T^\dagger_{\phi_j}\rangle$ and the product state $|T^\dagger_{\boldsymbol{\phi}}\rangle$ defined by:

$$|T^\dagger_{\boldsymbol{\phi}}\rangle := |T^\dagger_{\phi_1}\rangle \otimes \ldots \otimes |T^\dagger_{\phi_t}\rangle, \qquad |T^\dagger_{\phi_j}\rangle := T^\dagger_{\phi_j} H|0\rangle\,. \tag{3}$$

We use ξ^* to denote the quantity known as the *stablizer extent* [30] of the state $|T^\dagger_{\boldsymbol{\phi}}\rangle$, and we will formally define it later (see Eq. (41) in Sec. IV). For the moment, we simply note that for single qubit product states the stabilizer extent is multiplicative [30], i.e., ξ^* is a product of stabiliser extents $\xi(|T^\dagger_{\phi_j}\rangle)$ (so that ξ^* is a mild exponential in t), and that the extent of the single qubit state $\xi(|T^\dagger_{\phi}\rangle)$ is a simple function of ϕ that is upper bounded by ≈ 1.17.

Given some ordered subset $\mathcal{J} \subseteq [n]$ of w qubits to be measured in the computational basis and some outcome $x = (x_1, \ldots, x_w) \in \{0,1\}^w$, our aim is to compute or estimate the probability $p(\mathcal{J}, x)$ of observing the outcome x when measuring the final state $U|0\rangle^{\otimes n}$. Without loss of generality, we will assume that the first w qubits are measured and hence $\mathcal{J} = \{1, \ldots, w\}$. We refer to the first w qubits as the *measured* register 'a' and the remaining $(n-w)$ qubits as the *marginalized* register 'b'. Our central goal is to exactly or approximately compute the Born rule probability:

$$p := \left\| \langle x|_{\mathrm{a}} U|0\rangle^{\otimes n}_{\mathrm{ab}} \right\|_2^2. \tag{4}$$

The Born rule probability in the above equation can be described by specifying $n \in \mathbb{N}$, $w \in [n]$, $x \in \{0,1\}^w$ and an elementary description of an n-qubit unitary U. We will refer to this information as an *elementary description* of p.

B. Related research

As described in the Introduction, methods for Born rule probability estimation generally fall into two categories: one operating within the density state formalism, and the other focused on the setting of pure states.

In the density state formalism, algorithms known as quasi-probabilistic simulators [20–23] produce additive precision estimates of Born rule probabilities. These algorithms represent the quantum density state as a linear combination of a preferred set of operators known as a *frame* [23, 32]. Many frame choices have been considered including Weyl-Heisenberg displacement operators [18–20], frames constructed from stabilizer states [21, 22] and phase-point operators [23–25] used in the construction of the discrete Wigner function [33, 34]. Particularly relevant to our work is the dyadic frame simulator of Seddon *et al.* [22]. In this simulator, density states are decomposed into a linear combination of stabilizer dyads: operators of the form $|L\rangle\langle R|$ where $|L\rangle$ and $|R\rangle$ are pure stabilizer states. The efficiently simulable circuits consist of initial states that are a tensor products of convex combination of stabilizer dyads, stabilizer preserving operations including Clifford gates and computational basis measurements. These circuits can be promoted to universality by allowing initial states to include many copies of a magic state: states that are not a convex combination of stabilizer dyads and can be used to teleport non-Clifford gates into the circuit. The degree to which the initial state's optimal linear decomposition into stabilizer dyads departs from a convex combination is quantified by the *dyadic negativity*. The run-time of the dyadic frame simulators depends quadratically on the dyadic negativity. The dyadic negativity can in general be exponentially large and is the only source of run-time inefficiency. Nevertheless, in contrast to the Aaronson and Gottesman simulator, the dyadic frame simulator's run-time will be responsive to the level of deviation from the efficiently simulable operations. The dyadic frame simulator of Ref. [22] is the current state-of-the-art quasi-probabilistic simulator for simulating stabilizer circuits promoted to universality via magic state injection.

In the pure state formalism, a number of works [26–28] have culminated in two important simulation algorithms by Bravyi and Gosset (BG) [29]. The first of these, which we refer to as the *BG-estimation algorithm*, produces multiplicative precision estimates of Born rule probabilities. The second of these, which we refer to as the *BG-sampling algorithm*, approximately samples from the outcome distribution of the quantum circuit. These algorithms exactly or approximately represent the initial quantum state by a linear combination of stabilizer states. The efficiently simulable circuits consist of initial states that are a superposition of at most polynomially many stabilizer states, together with Clifford gates and computational basis measurements. These circuits can be promoted to universality by allowing initial states to include many copies of a magic state. The run-times of the BG-estimation and BG-sampling algorithms

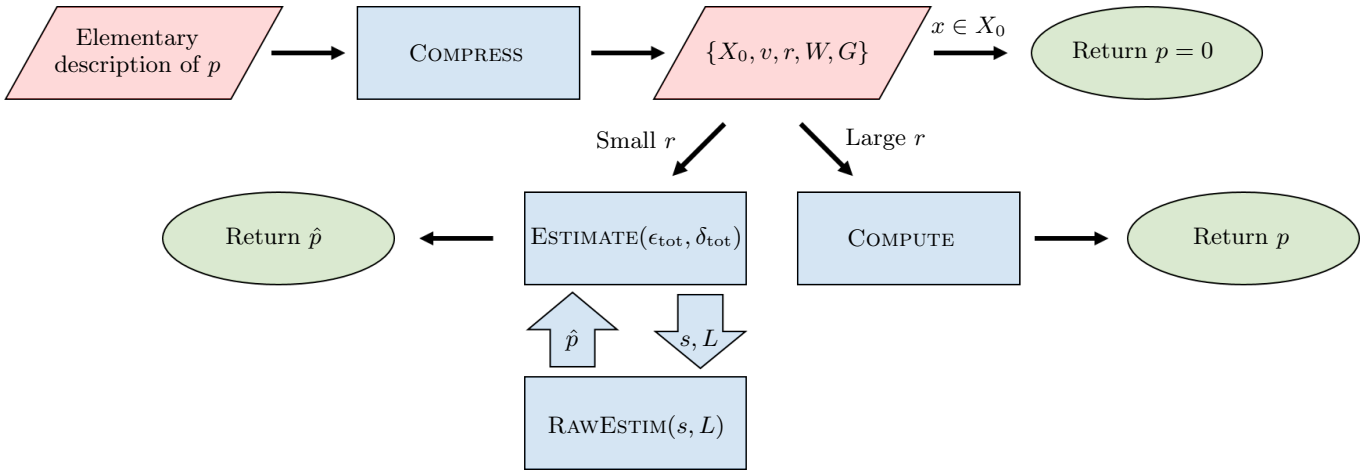


FIG. 1. **Flowchart for our main algorithms.** A component of the output of Compress is the identification of v measured qubits which have a deterministic outcome. Here, X_0 represents the set of measurement outcomes that are inconsistent with this deterministic outcome (also output by Compress). Owing to the Compute algorithm's run-time having an inverse exponential dependence on r (like 2^{t-r}), the choice of preferred algorithm is informed by the size of r (relative to t). The broad blue arrows indicate multiple calls by Estimate to RawEstim with different parameters s and L, and multiple outputs of $\hat{p}$ that are fed back to Estimate.

depend linearly on the exact and approximate stabilizer rank of the initial quantum state respectively. Roughly, the exact (or approximate) stabilizer rank of a quantum state is the minimal number, χ, of stabilizer states required such that this state can exactly (or approximately) be written as a linear combination of χ stabilizer states. Both algorithms have run-times that scale linearly in their respective stabilizer ranks and efficiently in all other circuit parameters, although some of the polynomial dependencies are nevertheless significant and can be prohibitive. Both the exact and approximate stabilizer ranks are computationally hard to compute even for product states although upper bounds exist for some important examples. Ref. [30] introduced a computationally better-behaved quantity ξ, called the stabilizer extent, and showed that the approximate stabilizer rank of an initial state $|\psi\rangle$ can be upper bounded by $\xi(|\psi\rangle)/\epsilon^2$, where ϵ quantifies the degree of error in the approximation of $|\psi\rangle$. Ref. [30] also presented the *sum over Cliffords* sampling algorithm: a new variant of the BG-sampling algorithm where non-Clifford gates are directly simulated by expressing them as a linear combination of Clifford gates. To compare this to our work, we consider the application of this technique to diagonal single qubit non-Clifford gates T_ϕ inducing a $Z-$rotation of angle $\phi \in (0, \pi/4)$. For circuits composed of exactly t uses of T_ϕ, the run-time of the sum over unitaries algorithm scales linearly in the stabilizer extent of the state $|T_\phi^\dagger\rangle$.

The mixed-state stabilizer rank simulator of Ref. [22] made further improvements to the BG-sampling algorithm by improving the run-time dependence on the error tolerance for the approximate sampling task and by generalizing the algorithm to the setting where initial states can be mixed states. The mixed-state stabilizer rank simulator's run-time scales linearly in a quantity known as the mixed state extent [22]. Ref. [22] also showed that for any $n-$qubit product states, its dyadic negativity, stabilizer extent and mixed state extent are all equal. This result allows one to compare performance across multiple simulation algorithms in the practically relevant setting where initial states are product states.

C. Summary of results

Our main results consist of two classical algorithms Compute and Estimate that either exactly compute or estimate the Born rule probability p given its elementary description. These make use of two auxiliary algorithms Compress and RawEstim, and the relation between them is presented in Fig. 1. The C+Python implementation of our algorithms used to generate Figures 2, 3 and 4 can be found in Ref. [35].

1. *The* Compress *algorithm*

The Compress algorithm is the starting point of our Born rule probability estimator. It takes as input the elementary description of p, and the purpose of this algorithm is to efficiently transform the elementary description of the Born rule probability into an alternative form, where we can decide what type of estimator is most suitable.

The Compress algorithm is composed of three steps, which we summarize here; a detailed description is presented in Sec. III. In the first step, we use a "reverse gadgetization" of T_{ϕ_j} gates (see Eq. (27)) to re-express the general circuit U acting on $|0\rangle_{\rm ab}^{\otimes n}$ as a Clifford circuit V acting on $|0\rangle_{\rm ab}^{\otimes n} \otimes |0\rangle_{\rm c}^{\otimes t}$, with the t ancillary qubits in register 'c' post-selected on the state $|T_{\boldsymbol{\phi}}^{\dagger}\rangle$. Thus, we re-express the Born rule probability from Eq. (4) as:

$$p = 2^t \left\| \langle x|_{\rm a} \langle T_{\boldsymbol{\phi}}^{\dagger}|_{\rm c} V |0\rangle_{\rm abc}^{\otimes n+t} \right\|_2^2. \tag{5}$$

Second, by imposing constraints on stabilisers, we re-express the Born rule probability p as:

$$p = 2^{v-w} \langle T_{\boldsymbol{\phi}}^{\dagger}| \prod_{i=1}^{t-r} (I + g_i) |T_{\boldsymbol{\phi}}^{\dagger}\rangle, \tag{6}$$

where $r \in \{0, 1, \ldots, \min\ \{t, n-w\}\}$ and $v \in \{0, 1, \ldots, w\}$ are circuit specific quantities (dependent on V) that are efficiently computable, and $\{g_i\}_{i=1}^{t-r}$ are t-qubit Pauli operators (generators of a stabiliser group). Finally, by explicitly constructing a gate sequence based on a stabiliser generator matrix, we re-express p as:

$$p = 2^{t-r+v-w} \left\| \langle 0|^{\otimes t-r} W |T_{\boldsymbol{\phi}}^{\dagger}\rangle \right\|_2^2, \tag{7}$$

where W is a t-qubit Clifford circuit of length $O(t^2)$ with $O(t)$ Hadamard gates.

The performance of Compress is captured by the following theorem:

Theorem 1 (Compress algorithm)**.** *Given an elementary description of p,* Compress *outputs the* deterministic number $v \in \{0, 1, \ldots, w\}$*, specifies a size v subset of the measured qubits that have deterministic outcomes and provides the measurement outcomes these qubits must produce. If the input x is consistent with these deterministic outcomes, then the algorithm outputs the* projector rank $r \in \{0, 1, \ldots, \min\ \{t, n-w\}\}$ *and an elementary description of the t-qubit Clifford unitary W together with a set G of $(t-r)$ Pauli operators g_i on t qubits such that:*

$$p = 2^{t-r+v-w} \left\| \langle 0|^{\otimes t-r} W |T_{\boldsymbol{\phi}}^{\dagger}\rangle \right\|_2^2 = 2^{v-w} \langle T_{\boldsymbol{\phi}}^{\dagger}| \prod_{i=1}^{t-r} (I + g_i) |T_{\boldsymbol{\phi}}^{\dagger}\rangle. \tag{8}$$

The run-time $\tau_{\rm Compress}$ of the algorithm scales as

$$\tau_{\rm Compress} = \mathrm{poly}(n, c, t). \tag{9}$$

The proof of this theorem is given in Sec. III C.

If x is not consistent with the deterministic outcomes specified by the Compress algorithm then we immediately conclude that $p = 0$ and we have efficiently calculated the target Born rule probability. Otherwise, we have two choices: either to use the Compute or the Estimate algorithm. In making this choice, the size of r relative to t will be important. The quantity r is the exponent of the rank of the stabilizer projector defined by G but we will call it the *projector rank* for short.

2. *The* Compute *algorithm*

If Compress outputs a large value of r relative to t (in the sense that $(t-r)$ is small), the Compute algorithm is likely to outperform our Estimate algorithm. Compute directly calculates 2^{t-r} terms appearing if one multiplies out the product appearing in Eq. (6). For each term we first need to calculate the product of $(t-r)$ Pauli operators of length t, and then compute the expectation value of product observables for product states of t qubits. Thus, the algorithm scales as $O\left(2^{t-r}(t-r)t\right)$, and projector rank r can be interpreted as the effective number of non-Clifford gates appearing in the original circuit. Full details of this algorithm are given in Sec. III.

The performance of Compute is captured by the following theorem:

Theorem 2 (COMPUTE algorithm)**.** *Given the output of the* COMPRESS *algorithm,* COMPUTE *outputs p (up to machine precision) in the run-time:*

$$\tau_{\text{COMPUTE}} = O\left(2^{t-r}(t-r)t\right). \tag{10}$$

The proof of this theorem is given in Sec. III B.

3. *The* ESTIMATE *algorithm*

If the projector rank is too small and τ_{COMPUTE} becomes infeasible, we may use our main result: the ESTIMATE algorithm. This algorithm produces Born rule probability estimates satisfying a desired additive error and failure probability. Our ESTIMATE algorithm makes use of a crucial subroutine we call RAWESTIM. This subroutine produces an estimate $\hat{p}$ of p given run-time constraints specified by a pair of parameters s and L. Optimal values for these parameters leading to estimates that satisfy a desired additive error and failure probability are determined by ESTIMATE. Here, we will first summarize the RAWESTIM subroutine (details of which are presented in Sec. IV), and then briefly describe our ESTIMATE algorithm (with details in Sec. V).

At its core, the RAWESTIM algorithm uses a concentration inequality (see Lemma 7) to bound the norm between a target vector $|\mu\rangle$ and a "simulated" approximation $|\overline{\psi}\rangle$. The target quantity p is directly related to the Euclidean norm of the target vector $|\mu\rangle$. Thus, an estimate of the Euclidean norm of the approximation vector $|\overline{\psi}\rangle$ is used to compute an estimate of p. The approximation vector $|\overline{\psi}\rangle$ is a uniform superposition of s randomly sampled stabilizer states. The sample space of stabilizer states and the probability distribution over these is directly constructed from stabilizer decompositions of magic states $|T_\phi^\dagger\rangle$. The RAWESTIM algorithm also uses a number of novel techniques to improve the run-time.

The RAWESTIM algorithm is composed of three steps, which are briefly summarized as follows. In the first step, we decompose the state $|T_\phi^\dagger\rangle$ appearing in Eq. (7) into a superposition of stabilizer states, thus re-expressing the Born rule probability p as the length $\||\mu\rangle\|_2^2$ of the following vector:

$$|\mu\rangle = \sum_y q(y)\,|\psi(y)\rangle\,. \tag{11}$$

Here, the sum is over all binary strings y of length t, $q(y)$ is a product probability distribution and $|\psi(y)\rangle$ are unnormalised stabiliser states on r qubits given by:

$$|\psi(y)\rangle \propto \langle 0|^{\otimes t-r} W\,|\tilde{y}\rangle\,, \tag{12}$$

where $|\tilde{y}\rangle$ is a t-fold tensor product of single qubit stabiliser states with $y_j = 0$ or $y_j = 1$ meaning that qubit j is in a stabiliser state $|+\rangle$ or $|-i\rangle$. We independently sample bit strings y, with probability $q(y)$, a total of s times, each time returning an r-qubit stabilizer state $|\psi_j\rangle$ equal to $|\psi(y)\rangle$ for the sampled y (the fast computation of $|\psi(y)\rangle$ is discussed in the next step). The uniform superposition of all s sampled stabilizer states $|\overline{\psi}\rangle$ is used to approximate $|\mu\rangle$. The distance between $|\mu\rangle$ and $|\overline{\psi}\rangle$ for a given s is sensitive to the lengths of $|\psi(y)\rangle$, which we upper-bound for all y using the stabilizer extent:

$$\max_y \||\psi(y)\rangle\|_2^2 \leq \xi^*. \tag{13}$$

In the second step, each sampled state $|\psi_j\rangle$ in the previous step is an unnormalised stabiliser state given by Eq. (12). We compute and represent these states in the phase sensitive CH form introduced in Ref. [30]. In order to obtain the needed CH forms of $|\psi_j\rangle$ we do the following. First, even before taking any samples, we pre-compute the CH form of $W\,|\tilde{0}\ldots\tilde{0}\rangle$ using the phase-sensitive simulator of Ref. [30]. Then, for each sampled y, we efficiently update the CH form of $W\,|\tilde{0}\ldots\tilde{0}\rangle$ to get the CH form of $W\,|\tilde{y}\rangle$. Finally, we use a novel subroutine that efficiently yields the CH form of the post-selected state $\langle 0|^{\otimes t-r} W\,|\tilde{y}\rangle$, and so of $|\psi(y)\rangle$. The vector $|\overline{\psi}\rangle$ is represented and stored as the CH forms of $|\psi_j\rangle$ for $j \in [s]$.

Finally, as the third step, we employ the fast norm estimation algorithm from Ref. [30] to estimate the norm of $|\overline{\psi}\rangle$. The square of the returned norm is the RAWESTIM algorithm's Born rule probability estimate $\hat{p}$.

The RAWESTIM algorithm's performance is characterized by the following theorem:

Theorem 3 (RawEstim algorithm)**.** *Given the output of the* Compress *algorithm and two positive integers s and L,* RawEstim *outputs an estimate $\hat{p}$ of the outcome probability p such that for all $\epsilon_{\text{tot}} > 0$ and $\epsilon \in (0, \epsilon_{\text{tot}})$:*

$$\Pr\left(|\hat{p} - p| \geq \epsilon_{\text{tot}}\right) \leq 2e^2 \exp\left(\frac{-s(\sqrt{p+\epsilon} - \sqrt{p})^2}{2(\sqrt{\xi^*} + \sqrt{p})^2}\right) + \exp\left(-\left(\frac{\epsilon_{\text{tot}} - \epsilon}{p+\epsilon}\right)^2 L\right) =: \delta_{\text{tot}}. \tag{14}$$

The run-time τ_{RawEstim} of the algorithm scales as

$$\tau_{\text{RawEstim}} = O\left(st^2(t-r) + sLr^3\right). \tag{15}$$

The proof of this theorem is presented in Sec. IV.

Given the output of the Compress algorithm and accuracy parameters $\epsilon_{\text{tot}}, \delta_{\text{tot}} > 0$, Estimate outputs an estimate $\hat{p}$ of the outcome probability p such that:

$$\Pr\left(|\hat{p} - p| \geq \epsilon_{\text{tot}}\right) \leq \delta_{\text{tot}}. \tag{16}$$

The RawEstim algorithm is used as a subroutine of the Estimate algorithm to achieve the desired error $\epsilon_{\text{tot}} > 0$ and failure probability $\delta_{\text{tot}} > 0$. With the proper choice of input parameters s and L, the RawEstim algorithm can achieve a desired failure probability δ_{tot} of the estimate $\hat{p}$. However, this proper choice depends on the unknown quantity p that we want to estimate. One could always make the conservative choice of $p = 1$ in Eq. (14), which will result in well-defined but highly suboptimal (too large) input parameters s and L. In contrast, the run-time of our Estimate algorithm takes advantage of improvements that become significant for small p. The Estimate algorithm achieves this by calling the RawEstim subroutine multiple times, with different choices of s and L. It starts with $s = s_0$ and $L = L_0$ so small that they cannot possibly satisfy the desired accuracy requirement. Then, at each step it chooses larger s_k, L_k that lead to estimates $\hat{p}_k$, which are used to learn upper bounds on p that decrease with each iteration. These, in turn, allow one to estimate sharper values of s and L to achieve the desired accuracy.

The run-time of Estimate, τ_{Estimate}, has two distinct components we call the *circuit-sensitive* and the *circuit-insensitive* components. The circuit-sensitive component of τ_{Estimate} is associated with the total run-time over all calls to the RawEstim subroutine. The run-time of the RawEstim subroutine will approximately double in each subsequent call with the run-time of each round and the total number of rounds depending on circuit parameters (such as t) and accuracy parameters (such as ϵ_{tot}). Typically, this component constitutes the overwhelming majority of τ_{Estimate}. The circuit-insensitive component of τ_{Estimate} arises from various numerical optimizations that are executed in each step of the Estimate algorithm, e.g. to determine the choice of s_k, L_k for each step k. The run-time of each such step is of order ~ 1 second (for a standard desktop computer) and it is insensitive to the various parameters that define the Born rule probability estimation task. The total number of steps is also small with more than ~ 50 steps being infeasible due to the exponential growth of the run-time of RawEstim in the step number k. For this reason, we treat the circuit-insensitive component of τ_{Estimate} as a fixed run-time cost.

Consistent with Eq. (15), we model the run-time of RawEstim as:

$$\tau_{\text{model}}(s, L) := c_1 st^2(t-r) + c_2 sLr^3, \tag{17}$$

where c_1, c_2 are hardware specific positive constants (in units of seconds per elementary operation) that can be used to model the actual run-time of RawEstim. The Estimate algorithm aims to minimize the quantity:

$$\mathcal{C} := \sum_{k \in [K]} \tau_{\text{model}}(s_k, L_k), \tag{18}$$

where K is the total number of times the RawEstim algorithm will be called and s_k, L_k indicate the input parameters used on the k^{th} call. We call $\mathcal{C}$ the *run-time cost*; it represents our modelled circuit-sensitive component of the run-time of Estimate.

The run-time cost $\mathcal{C}$ is probabilistic and depends on the unknown p. Our RunTime algorithm efficiently computes a probabilistic upper bound of $\mathcal{C}$ for any assumed p. This may be useful for informing expected run-times particularly when prior information about p is known. Our Estimate and RunTime algorithms, together with related details, can be found in Sec. V. We note that our Estimate algorithm allows the user to fix the accuracy parameters, ϵ_{tot} and δ_{tot}, for the price of moving their dependence on p to τ_{Estimate}.

D. Discussion of the performance of our algorithms

Our suite of algorithms offer state-of-the-art performance in Born rule probability estimation across a broad range of parameter regimes, as we now describe.

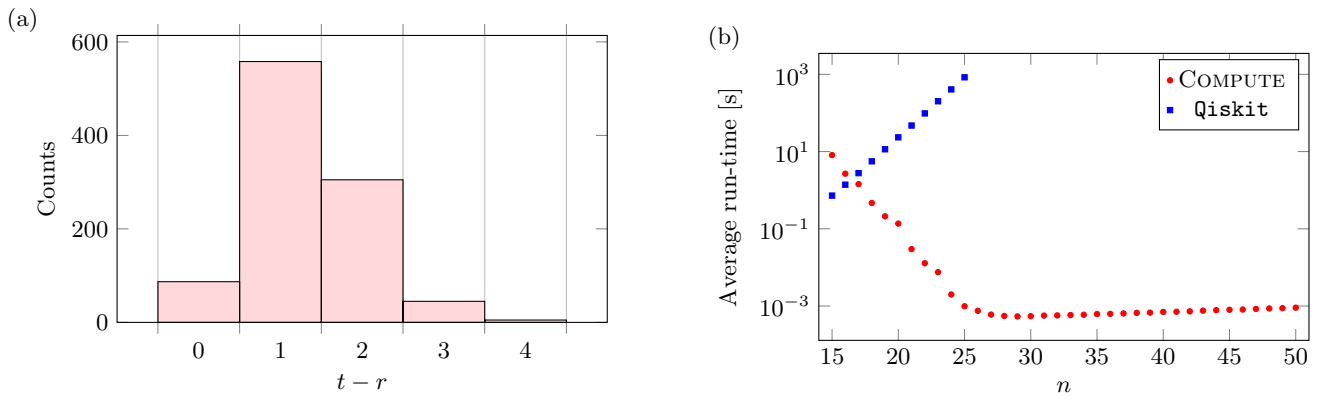


FIG. 2. **Performance of the COMPUTE algorithm for random circuits.** Random circuits are generated as follows: we generate c Clifford gates acting on random qubits (equal probability of choosing S, H, CX and CZ), and then replace randomly selected t of them with T gates. (a) The distribution of the compressed T-count $(t-r)$ for 10^3 random circuits with $n = 100$ qubits, $c = 10^5$ Clifford gates, $t = 80$ T gates and $w = 20$ measured qubits. (b) Average run-times for calculating the Born rule probability for random circuits with n qubits, $c = 10^3$ Clifford gates, $t = 30$ T gates and $w = 10$ measured qubits, with the average taken over 10^2 random circuits for each n. The red circles correspond to our COMPUTE algorithm (including the run-time needed to run COMPRESS), while the blue squares correspond to classical state vector simulation framework of IBM's quantum programming suite `Qiskit` [36]. Simulations were performed on a standard desktop computer.

We first discuss the performance of the COMPUTE algorithm, noting that it depends exponentially on $(t-r)$. In the case of random circuits where many Clifford gates are interleaved between each non-Clifford gate, our numerical investigations show that r very strongly concentrates around the maximum allowed value of $\min\{t, n-w\}$, see Fig. 2a for details. Thus, in certain parameter regimes, e.g., when $(n-w) \geq t$, the COMPUTE algorithm has a very quick run-time. In Fig. 2b we present the comparison of the run-times between our COMPUTE algorithm and the IBM's `Qiskit` state vector simulator [36]. While the run-times for the latter algorithm become infeasible on a standard desktop computer for $n > 35$ (due to memory limitations), our algorithm can, within feasible runtimes, compute the Born rule probabilities as long as the number of non-Clifford gates t is not significantly larger than $(n-w)$. Thus, for random circuits it is not the total number of non-Clifford gates that makes our simulation infeasible, but rather the number of non-Clifford gates in excess of the number of unmeasured qubits. To illustrate this, we employed COMPUTE to get the Born rule probability of a random circuit with $n = 55$, $w = 5$, $c = 10^5$ and $t = 80$, and the total run-time was 5586 seconds. For this circuit, r was found to take its maximal value $r = 50$. By randomly sampling $1,000$ such circuits and processing them through COMPRESS we found that r was exactly 50 in all cases. Hence our COMPUTE run-time for such circuits is typical.

Let us also compare our COMPUTE algorithm with the BG-estimation algorithm [29]. There are two obvious benefits of our algorithm. First, we are not limited to T gates but allow for general diagonal gates T_ϕ, which can significantly reduce the run-time [37]. This is because a diagonal gate T_ϕ with a small angle ϕ requires many T gates to be synthesised (which increases the simulation cost), while we can use a single T_ϕ gate in our simulator. Second, our algorithm is exact, while the one of Ref. [29] runs with a failure probability δ and relative error ε, and to improve these precision parameters one has to pay the price of longer run-times. Specifically, the run-time of that algorithm is given by $O\left(2^{\beta t} t^3 \varepsilon^{-2} \log(\delta^{-1})\right)$, where $\beta = (1/6)\log_2 7 \approx 0.47$. Comparing this with τ_{COMPUTE}, we see that the performance of our algorithm is better in certain parameter regimes when $(t-r) \leq \beta t$. As discussed above, this happens generically for random circuits when $(1-\beta)t \leq n-w$.

The analysis of the performance of ESTIMATE will be divided into three parts. First, we will discuss the crucial RAWESTIM subroutine and point out the run-time improvements over the existing Born rule estimation algorithms. Second, we will explain additional run-time improvements that arise from the ESTIMATE algorithm itself, i.e., from the adaptative choice of optimal input parameters s and L for the RAWESTIM subroutine that leads to upper bounds on the estimate value of p. Finally, we will explain why we expect the total run-time of ESTIMATE to be closely related (to within 1-2 orders of magnitude) to the run-time of RAWESTIM with the optimal choice of parameters; and we will also provide numerical evidence supporting these expectations. The performance of our ESTIMATE algorithm is illustrated in Fig. 3.

To analyse the performance of RAWESTIM, we start by employing Eq. (14) to note that for arbitrary $\epsilon \in (0, \epsilon_{\text{tot}})$

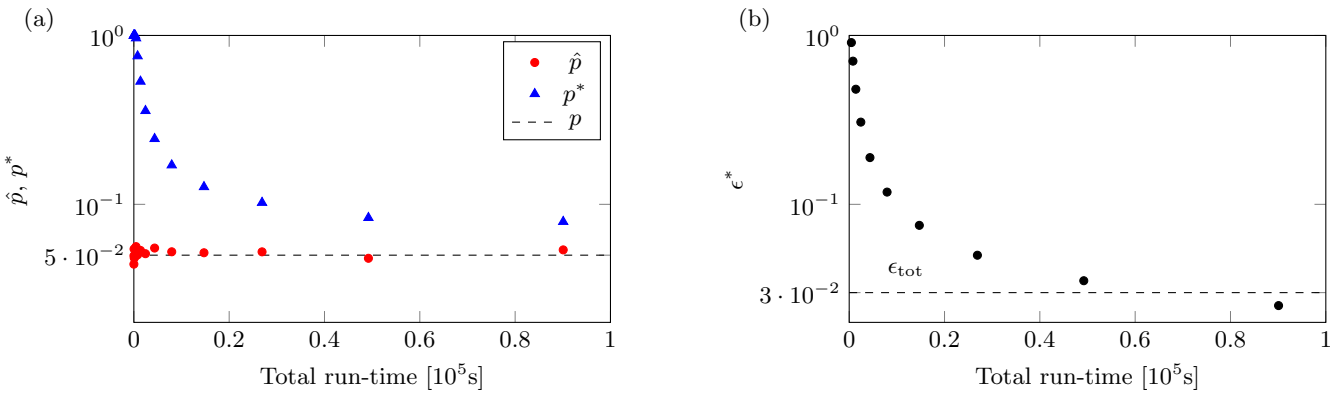


FIG. 3. **Performance of the** ESTIMATE **algorithm.** An $n = 50$ qubit, $t = 60$ non-Clifford gate circuit of the form $UU^\dagger V(p)$ as described at the end of Sec. II D. The unitary U is randomly constructed as described in the caption of Fig. 2, and consists of 1000 gates of which 26 are non-Clifford T_θ gates. The unitary $V(p)$ acts non-trivially on the first $w = 8$ qubits, which are then measured in the computational basis, leading to the the probability of the all-zero outcome $p = 0.05$. The parameter θ is chosen such that the total circuit has stabiliser extent $\xi^* \approx 3767$, equivalent to 52 T gates. For this circuit the value of projector rank is $r = 10$. The total run-time of approximately 9×10^4s includes approximately 38s of fixed overhead from the ESTIMATE algorithm. (a) The estimate $\hat{p}$ (red circles) and its upper bound p^* (blue triangles) as a function of the total run-time. The dashed line indicates the chosen value of p. (b) The upper-bound ϵ^* for the total estimation error as a function of the total run-time. The dashed line indicates the target error $\epsilon_{\text{tot}} = 0.03$, and a failure probability of $\delta_{\text{tot}} = 10^{-3}$ was used.

and $\delta \in (0, \delta_{\text{tot}})$ the choice of parameters s and L satisfying

$$s \geq \frac{2(\sqrt{\xi^*} + \sqrt{p})^2}{\left(\sqrt{p+\epsilon} - \sqrt{p}\right)^2} \log\left(\frac{2e^2}{\delta}\right), \qquad L \geq \left(\frac{p+\epsilon}{\epsilon_{\text{tot}} - \epsilon}\right)^2 \log\left(\frac{1}{\delta_{\text{tot}} - \delta}\right), \tag{19}$$

guarantees an estimate $\hat{p}$ with error smaller than ϵ_{tot} and failure probability smaller than δ_{tot}. The meaningful parameter regime is given by $\epsilon_{\text{tot}} \ll p$ (estimation error should be smaller than the estimated value) and $\xi^* \gg 1$ (we want to simulate non-Clifford circuits, as Clifford ones are already efficiently simulable). Then, the two terms of the run-time τ_{RAWESTIM} characterized by Eq. (15) scale as

$$\tau^{(1)}_{\text{RAWESTIM}} = \tilde{O}\left(\xi^* t^2 (t-r) p \epsilon_{\text{tot}}^{-2}\right), \qquad \tau^{(2)}_{\text{RAWESTIM}} = \tilde{O}\left(\xi^* r^3 p^3 \epsilon_{\text{tot}}^{-4}\right), \tag{20}$$

where $\tilde{O}$ notation hides the logarithmic dependence on the failure probability δ_{tot}. Importantly, note that the relative error ε_{tot} introduced by the additive error ϵ_{tot} is given by $\varepsilon_{\text{tot}} = \epsilon_{\text{tot}}/p$. Thus, the run-time only weakly depends on the additive error as $O\left(\epsilon_{\text{tot}}^{-1}\right)$ for both terms, with the remaining scaling dependent on the relative error as $O\left(\varepsilon_{\text{tot}}^{-1}\right)$ and $O\left(\varepsilon_{\text{tot}}^{-3}\right)$, respectively.

We first compare the performance of RAWESTIM with the results of Ref. [30], where the authors develop a state-of-the-art simulator that samples from the Born rule probability distribution. But they also provide a subroutine that, like our RAWESTIM algorithm, approximates Born rule probabilities to additive polynomial precision. This estimation algorithm is based on the approximate stabiliser decomposition of magic states and on a novel fast norm estimation subroutine. First, one computes k-rank stabiliser decomposition taking $O\left(kt^3\right)$ steps. The crucial Theorem 1 of Ref. [30] proves that by choosing $k \approx \xi^*/\epsilon_1^2$, the additive error introduced in this step will be bounded by ϵ_1. Next, one uses the fast norm estimation with a failure probability δ_{tot} and a relative error ε_2, which takes $\tilde{O}\left(kt^3\varepsilon_2^{-2}\right)$ steps, and the run-time of this step dominates the total run-time. Note that the worst case total additive error ϵ_{tot} can be lower-bounded by $\epsilon_1 + p\varepsilon_2$. Thus, the term $O\left(\epsilon_1^{-2}\varepsilon_2^{-2}\right)$ can be optimally replaced by $O\left(p^2\epsilon_{\text{tot}}^{-4}\right)$. Taking this into account, one gets that the total run-time is $\tilde{O}\left(\xi^* t^3 p^2 \epsilon_{\text{tot}}^{-4}\right)$. A variation of this algorithm, *the sum over Cliffords* method [30], has the total run-time of $\tilde{O}\left(\xi^* n^3 p^2 \epsilon_{\text{tot}}^{-4}\right)$. We combine and compare to the best of these performances given by $\tilde{O}\left(\xi^* \min\left\{n^3, t^3\right\} p^2 \epsilon_{\text{tot}}^{-4}\right)$. Comparing this with $\tau^{(1)}_{\text{RAWESTIM}}$ and $\tau^{(2)}_{\text{RAWESTIM}}$ (and noting that $r \leq t$, $r \leq n$, $p \leq 1$ and $\epsilon_{\text{tot}}^2/p \leq 1$), we see that RAWESTIM compares favourably in almost all regimes. More precisely, there is a performance advantage scaling as $\tilde{O}\left(p\epsilon_{\text{tot}}^{-2} \min\{1, (n/t)^3\}\right)$ and $\tilde{O}\left(p^{-1} \min\{(n/r)^3, (t/r)^3\}\right)$ for the two components of the run-time. Moreover, note that in the regime where $(t-r)$ is small, the performance of COMPUTE should be much better than that of ESTIMATE. Thus, a more natural regime for the RAWESTIM algorithm is when $r \ll t$, i.e., when the dominant run-time comes from $\tau^{(1)}_{\text{RAWESTIM}}$ component. The run-time improvement related to the estimated

probability and its error is then of the order $O\left(p\epsilon_{\text{tot}}^{-2}\right)$, so that the advantage becomes particularly significant for high accuracy estimates.

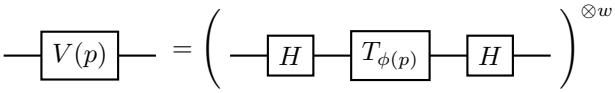
Next, we compare the performance of RawEstim with the results of Ref. [22]. We start by noting that the *mixed-state stabilizer rank* simulator of Ref. [22] improved the run-time by a factor of up to $\epsilon_{\text{tot}}^{-1}$ as compared to the sampling based simulation of Ref. [30]. This should be contrasted with our improvement factors of p^{-1} and $\epsilon_{\text{tot}}^{-2}/p$, and so, depending on the regime, the mixed-state stabilizer rank simulator could be better or worse than RawEstim. However, it should be noted that the improvement in Ref. [22] applies specifically to the task of approximately sampling from the outcome distribution of a quantum circuit. Therefore, it is unclear how to attain such an improvement directly for the task of Born probability estimation (we note that one can attain Born probability estimates by using $O\left(\epsilon_{\text{tot}}^{-2}\right)$ samples but this invalidates the run-time advantage). Reference [22] also presents the *dyadic frame simulator*. It performs exactly the same task as Estimate, i.e., it estimates a single Born rule probability with an additive error ϵ_{tot}, and we note that the dyadic frame simulator is more generally applicable as it is also suitable for mixed states. Ignoring the polynomial and logarithmic pre-factors, its dominant run-time scales as $O\left(\xi^{*2}\epsilon_{\text{tot}}^{-2}\right)$. Therefore, we see that our RawEstim algorithm compares favorably, as it has a run-time advantage of ξ^* that is exponential in the number t of non-Clifford gates.

We now proceed to discussing the second source of performance advantage that arises from the adaptive nature of the Estimate algorithm. In order to produce a meaningful estimate, we require guarantees on its error ϵ_{tot} and failure probability δ_{tot}. We note that neither RawEstim nor any of the above mentioned competing algorithms have such an accuracy guarantee, as in order to choose proper simulation parameters (like our s and L), achieving given ϵ_{tot} and δ_{tot}, one would need to know the unknown value of p. Thus, one is left to make a conservative choice of $p = 1$ that kills any run-time advantage coming from the polynomial dependence on p. On the other hand, our Estimate algorithm is able to take advantage of this p dependence. As a result, the run-time improvements related to the estimated probability and its error effectively scale as $O\left(p^{-1}\epsilon_{\text{tot}}^{-2}\right)$ and $O\left(p^{-3}\right)$ (rather than the above-mentioned $O\left(p\epsilon_{\text{tot}}^{-2}\right)$ and $O\left(p^{-1}\right)$). The run-time price of using Estimate, as compared to RawEstim with optimally chosen parameters s and L, is a small circuit-insensitive overhead related to parameter optimisation, and an additional circuit-sensitive overhead arising from the fact that we make multiple calls to RawEstim. The former one is so small that can be ignored, while we explain how to effectively upper-bound the latter one below. To conclude, Estimate exhibits the following run-time improvements as compared to the run-time $\tau_{[30]}$ of the two methods of Ref. [30]:

$$\frac{\tau_{[30]}}{\tau^{(1)}_{\text{RawEstim}}} = \tilde{O}\left(p^{-1}\epsilon_{\text{tot}}^{-2}\,\min\{1,(n/t)^3\}\right), \qquad \frac{\tau_{[30]}}{\tau^{(2)}_{\text{RawEstim}}} = \tilde{O}\left(p^{-3}\,\min\{(n/r)^3,(t/r)^3\}\right). \tag{21}$$

Finally, we will now explain why we expect that $\tau_{\text{model}}(s^*, L^*)$, with (s^*, L^*) being the choice of parameters s and L optimized with respect the unknown p, can act as a proxy for τ_{Estimate} in the regime where $p \geq \epsilon_{\text{tot}}$. The Estimate algorithm runs the RawEstim subroutine K times, at each step k, the parameters s_k and L_k are chosen optimally with respect to p_k^{UB}, an upper bound for p. It can be shown that in the final step, $p_K^{\text{UB}} \leq p + 2\epsilon_{\text{tot}}$. Thus, in the regime where $p \geq \epsilon_{\text{tot}}$, the optimization is with respect to $p_K^{\text{UB}} = O(p)$ with $\tau_{\text{model}}(s^*, L^*)$ having a cubic dependence on p. An additional source of discrepancy arises since the final step's optimisation uses a failure probability of $\delta_k = \frac{6}{\pi^2 K^2}\delta_{\text{tot}}$ in contrast to δ_{tot} used in determining s^* and L^*. However, due to $\tau_{\text{model}}(s^*, L^*)$ having only a poly-logarithmic dependence on δ_{tot}, this also contributes a small run-time overhead to the final step's call to RawEstim. Finally, since the final call's cost is approximately half of the total run-time cost we expect that run-time of Estimate to be within 1-2 orders of magnitude of τ_{RawEstim} when $p \geq \epsilon_{\text{tot}}$.

In order to verify our expectations, we performed the following analysis. We constructed quantum circuits of the form $UU^\dagger V(p)$, where U is a random non-Clifford circuit composed of Clifford and T_θ gates, and $V(p)$ is a non-Clifford circuit that acts non-trivially on the first w measured qubits as:



(22)

This way we were able to generate random non-Clifford circuits with a chosen probability $p \in [0, 1]$ of the all zero outcome controlled by the choice of parameter $\phi(p)$, and a stabilizer extent ξ^* that is made independent of p by controlling θ. Then, using the RunTime algorithm, we found the upper-bound of run-time cost $\mathcal{C}$ of the Estimate algorithm as a function of p. We have also lower-bounded this cost by the run-time cost of RawEstim with the optimal choice of s and L (as we know the value of p this can be easily done using Theorem 3). We present both bounds in Fig. 4, where it is clear that they differ by less than 2 orders of magnitude. To further strengthen our point, we have also run the Estimate algorithm on circuits $UU^\dagger V(p)$ for a few chosen values of p, and also plotted the actual run-time costs in Fig. 4. This shows that, provided $p \geq \epsilon_{\text{tot}}$, $\mathcal{C}$ is indeed close to the run-time cost of RawEstim with the choice of s and L that being optimised using knowledge of the value of p.

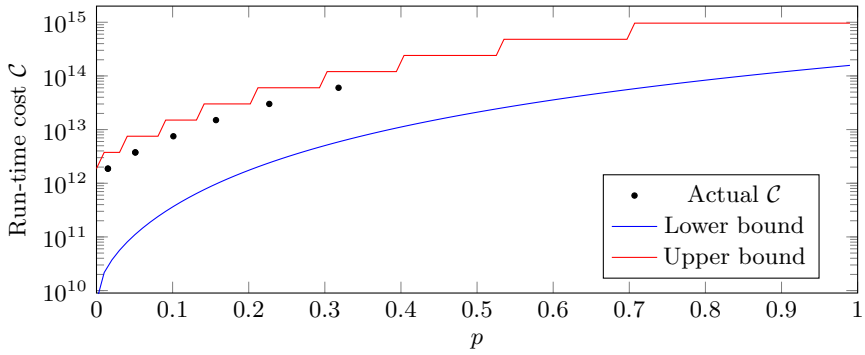


FIG. 4. **Estimate run-time cost and its bounds.** The run-time cost of the ESTIMATE algorithm with $\epsilon_{\rm tot} = 0.05$, $\delta_{\rm tot} = 0.001$ for random circuits $UU^\dagger V(p)$. The black dots represent the total run-time cost $\mathcal{C}$, as defined in Eqs. (17)-(18) using $c_1 = c_2 = 1$. The circuits were acting on $n = 40$ qubits and were composed of $t = 40$ non-Clifford gates with the total stabilizer extent $\xi^* = 158.715$ (equivalent to 32 T gates). Each black dot is in fact a cluster of 2-3 independent ESTIMATE simulations that produce $\mathcal{C}$ values that are too close to resolve on this plot. All final Born rule probability estimates produced by ESTIMATE were within an additive error $0.2\epsilon_{\rm tot}$ of p. The top red line indicates the probabilistic upper bound (with failure probability less than $\delta_{\rm UB} = 0.05$) for the total run-time cost $\mathcal{C}$ obtained with the efficient RUNTIME algorithm. The bottom blue line indicates the lower bound on $\mathcal{C}$ obtained from $\tau_{\rm RAWESTIM}$ with the choice of parameters s and L being optimized using knowledge of the value of p. The run-time cost of the first data point with $p = 0.015$ corresponds to actual computational time of 478 seconds.

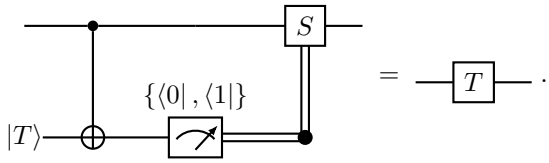
III. THE COMPRESS AND COMPUTE ALGORITHMS

A. Step 1: Gadgetization

It is well known that a T gate acting on a given qubit can be replaced by its gadgetised version [38, 39]. More precisely, one can prepare an ancillary qubit in a magic state

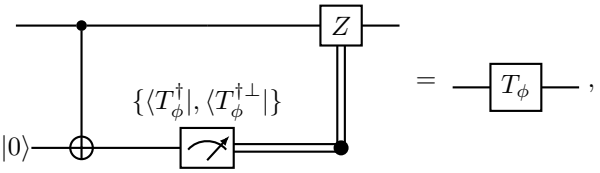
$$|T\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \exp(i\pi/4)\,|1\rangle), \tag{23}$$

couple it to the original qubit by a CX gate (with the original qubit acting as the control) and measure in the computational basis. Then, if the outcome is $|1\rangle$, one also needs to apply a correction Clifford phase gate S to the original qubit. The effect of the above procedure is the same as direct application of the T gate to a given qubit. Diagrammatically,



(24)

Here, we will employ an alternative construction that replaces the ancillary non-stabiliser $|T\rangle$ state with a non-Clifford measurement, and allows one to implement any diagonal T_ϕ gate. Our reverse gadget is obtained as follows:



(25)

with

$$|T_\phi^\dagger\rangle = \frac{1}{\sqrt{2}}(|0\rangle + \exp(-i\phi)\,|1\rangle), \quad |T_\phi^{\dagger\perp}\rangle = \frac{1}{\sqrt{2}}(|0\rangle - \exp(-i\phi)\,|1\rangle). \tag{26}$$

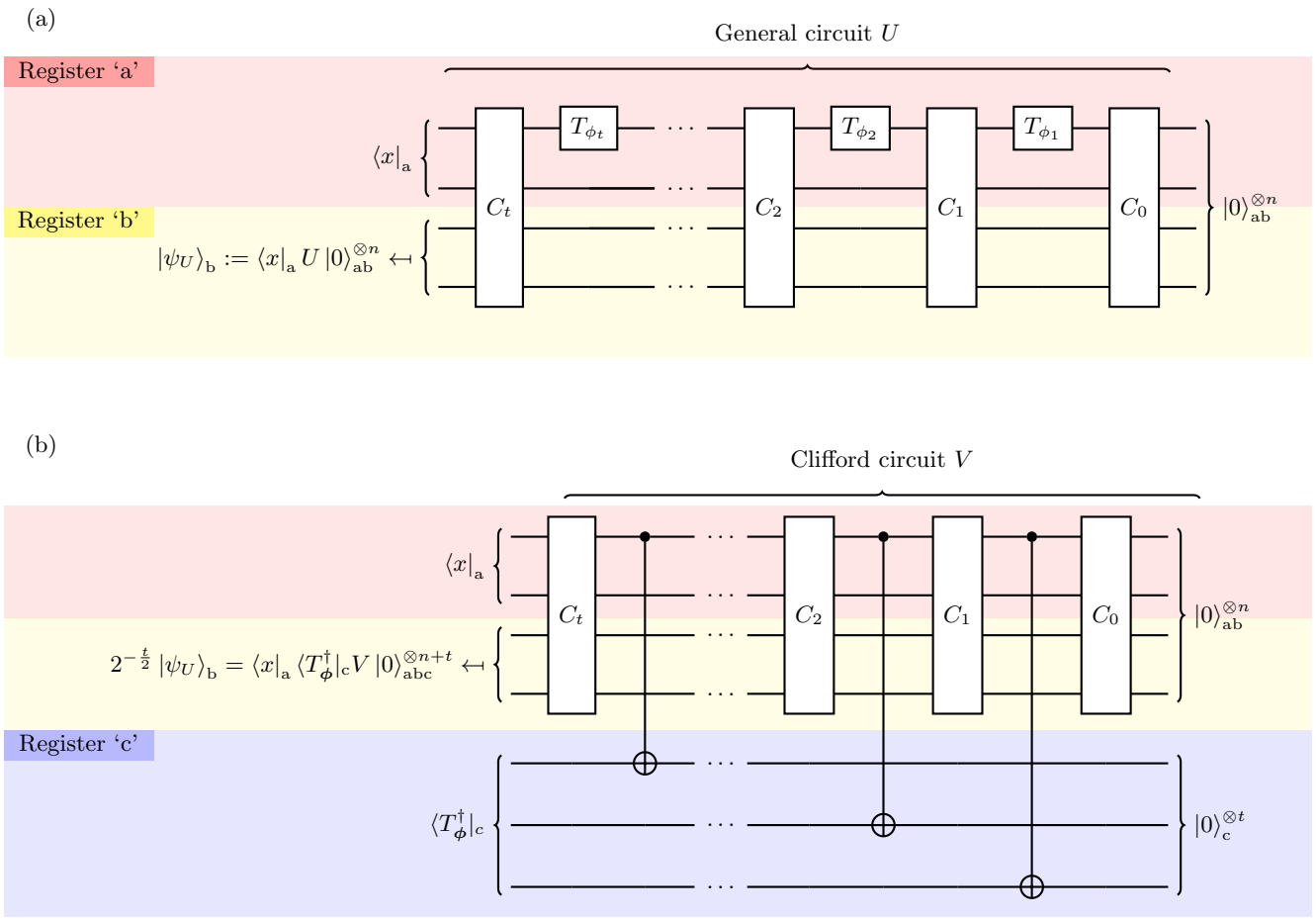
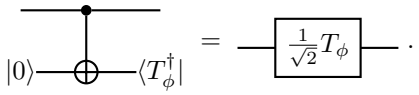


FIG. 5. **Circuits and registers.** Circuit diagrams should be read from right to left. (a) General circuit U composed of Clifford gates and t diagonal T_{ϕ_j} gates, post-selected on the x outcome of w qubits in register 'a'. Gates C_j consist of all Clifford gates appearing before the j-th diagonal gate T_{ϕ_j}. Note that non-Clifford diagonal gates act only on the first qubit for clarity of the figure and without loss of generality (since $SWAP$ gates are Clifford, each T_{ϕ_j} gate can effectively act on any qubit). (b) Post-selected circuit obtained by reverse gadgetisation of U. The unitary V is obtained from U by replacing each T_{ϕ_j} with a CX gate between the original qubit and an ancillary qubit in register 'c'. Qubits in register 'c' are then post-selected on the non-stabiliser state $|T_{\boldsymbol{\phi}}^{\dagger}\rangle$, while qubits in register 'b' are post-selected on the same outcome as in the original U circuit.

Next, it is straightforward to show that in the above reverse gadget the measurement outcomes of the ancillary qubit are equally likely. Therefore, we can focus on the $|T_{\phi}^{\dagger}\rangle$ outcome (as no correction gates are then needed), and consider a simplified post-selected circuit:



(27)

Now, for a circuit U consisting of c Clifford gates and t diagonal gates $\{T_{\phi_j}\}$, we can gadgetise each of the t occurrences of the non-Clifford gate in the way described above. Hence, we can replace a general unitary circuit U on n qubits in a state $|0\rangle_{\mathrm{ab}}^{\otimes n}$ by a Clifford circuit V on $n+t$ qubits in a state $|0\rangle_{\mathrm{abc}}^{\otimes n+t}$, which is post-selected on $|T_{\boldsymbol{\phi}}^{\dagger}\rangle$ outcome on the ancillary qubits in register 'c'. The unitary V is composed of $c+t$ Clifford gates: the original c Clifford unitaries appearing in the decomposition of U into Cliffords and non-Clifford gates, plus t instances of CX gates between computational and ancillary qubits arising from the reverse gadgetisation of T_{ϕ_i} gates. We illustrate this in Fig. 5, where we also present the division of all $n+t$ qubits into 3 registers: the measured register 'a' that we post-select on the $|x\rangle$ outcome, the marginalised register 'b', and register 'c' consisting of the ancillary qubits that we post-select on the $|T_{\boldsymbol{\phi}}^{\dagger}\rangle$ outcome. Due to the fact that all measurement outcomes in reverse gadgets are equally

probable, such a post-selected circuit V will realise U up to a renormalization factor:

$$U\left|0\right\rangle_{\mathrm{ab}}^{\otimes n} = 2^{t/2}\langle T_{\boldsymbol{\phi}}^{\dagger}|_{\mathrm{c}} V\left|0\right\rangle_{\mathrm{abc}}^{\otimes n+t}. \tag{28}$$

The probability of observing outcome x is thus given by

$$p = 2^{t}\left\|\langle x|_{\mathrm{a}}\langle T_{\boldsymbol{\phi}}^{\dagger}|_{\mathrm{c}} V\left|0\right\rangle_{\mathrm{abc}}^{\otimes n+t}\right\|_2^2. \tag{29}$$

The process of constructing V given an elementary description of U obviously has a polynomial run-time $\mathrm{poly}(n,c,t)$.

B. Step 2: Constraining stabilisers

In this step we will use the stabilizer formalism introduced in Refs [13, 16] to rewrite the expression for p given in Eq. (29) in a simplified form. It will lead directly to the Compute algorithm, and will be further simplified in the next step before serving as an input to the RawEstim algorithm. Moreover, we will also extract the crucial parameters describing the circuit V: the projector rank r and the deterministic number v. The first one of these effectively characterizes how much the number t of diagonal gates can be compressed, while the latter one is related to the number of outcomes with a zero probability.

Let us first briefly introduce some notation and recall standard techniques within the stabilizer formalism. An n-qubit Pauli operator, P, is any operator of the form $\omega P_1\otimes\ldots\otimes P_n$ where $\omega\in\{\pm1,\pm i\}$ and $P_j\in\{I,X,Y,Z\}$ are single qubit Pauli operators. We denote the set of all n-qubit Pauli operators by $\mathcal{P}_n$. For any $P\in\mathcal{P}_n$ and $j\in[n]$, we use $|P|_j$ to denote the j^{th} tensor factor P_j and $\omega(P)$ to denote the phase factor ω. We will slightly abuse notation by using $|P|_{\mathrm{a}}$ to denote the sub-string of tensor factors associated with register 'a', i.e. $|P|_{\mathrm{a}} := \otimes_{j\in[w]}P_j$ and similarly for $|P|_{\mathrm{b}}$ and $|P|_{\mathrm{c}}$.

We say that $P\in\mathcal{P}_n$ stabilizes an n-qubit quantum state $|\psi\rangle$ if and only if $P|\psi\rangle=|\psi\rangle$. The subset $S(|\psi\rangle)\subset\mathcal{P}_n$ consisting of all stabilizers of $|\psi\rangle$ is an Abelian group isomorphic to $\mathbb{Z}_2^n$. This group can be non-uniquely represented by a generator set $G=\{g_1,\ldots,g_n\}\subset S(|\psi\rangle)$ such that $S(|\psi\rangle)=\langle G\rangle$. For $k\leq n$, $G=\{g_1,\ldots,g_k\}$ is an n-qubit, k-element generating set if and only if $g_i\in\mathcal{P}_n$ for all $i\in[k]$, all pairs $g_i,g_j\in G$ commute and G is independent, i.e. for all $i\in[k]$, $g_i\notin\langle G\setminus\{g_i\}\rangle$. We denote the set of all n-qubit, k-element generating sets by $\mathcal{G}(n,k)$. For $G=\{g_1,\ldots,g_k\}\in\mathcal{G}(n,k)$ we define the associated projector:

$$\Pi_G := \prod_{i=1}^{k}\frac{I+g_i}{2} \tag{30a}$$

$$= 2^{-k}\sum_{g\in\langle G\rangle} g. \tag{30b}$$

We can now state the crucial lemma of this step. Its rigorous proof including the pseudo-code of the algorithm and associated sub-procedures can be found in Appendix A. Here, we will limit ourselves to a high level description of the main idea behind the proof.

Lemma 4 (ConstrainStabs algorithm)**.** *Given an elementary description of p,* ConstrainStabs *outputs deterministic number $v\in\{0,1,\ldots,w\}$, projector rank $r\in\{0,1,\ldots,\min\ \{t,n-w\}\}$, a set $J=\{j_1,\ldots,j_v\}\subseteq[w]$, a bitstring $x'=(x'_1,\ldots,x'_v)$ and two generating sets $\tilde{G}\in\mathcal{G}(n+t,t-r+v)$ and $G\in\mathcal{G}(t,t-r)$ such that:*

$$\mathrm{Tr}_{\mathrm{ab}}\left(V\left|0\right\rangle\!\left\langle 0\right|_{\mathrm{abc}}^{\otimes n+t}V^{\dagger}\left|x\right\rangle\!\left\langle x\right|_{\mathrm{a}}\right) = 2^{-n-r+v}\mathrm{Tr}_{\mathrm{ab}}\left(\Pi_{\tilde{G}}\left|x\right\rangle\!\left\langle x\right|_{\mathrm{a}}\right) \tag{31a}$$

$$= 2^{-r+v-w}\Pi_G, \tag{31b}$$

and for all $k\in[v]$, $x_{j_k}\neq x'_k$ immediately implies $\Pi_G=0$. The run-time of the ConstrainStabs *algorithm is polynomial in the relevant parameters:*

$$\tau_{\textsc{ConstrainStabs}} = \mathrm{poly}(c,n,t). \tag{32}$$

Using Eq. (29), we note that Lemma 4 immediately implies that we can rewrite the Born rule probability p in the following two ways:

$$p = 2^{-n+t-r+v}\mathrm{Tr}\left(\Pi_{\tilde{G}}\left|x\right\rangle\!\left\langle x\right|_{\mathrm{a}}\otimes I_{\mathrm{b}}^{\otimes n-w}\otimes\left|T_{\boldsymbol{\phi}}^{\dagger}\right\rangle\!\left\langle T_{\boldsymbol{\phi}}^{\dagger}\right|_{\mathrm{c}}\right) \tag{33a}$$

$$= 2^{t-r+v-w}\mathrm{Tr}\left(\Pi_G\left|T_{\boldsymbol{\phi}}^{\dagger}\right\rangle\!\left\langle T_{\boldsymbol{\phi}}^{\dagger}\right|\right) = 2^{v-w}\langle T_{\boldsymbol{\phi}}^{\dagger}|\prod_{i=1}^{t-r}(I+g_i)|T_{\boldsymbol{\phi}}^{\dagger}\rangle, \tag{33b}$$

where in the last equality the product is over all $g_i \in \langle G \rangle$. Moreover, since for all $k \in [v]$, $x_{j_k} \neq x'_k$ immediately implies $\Pi_G = 0$, it also implies $p = 0$. Most importantly, by directly calculating all the terms appearing in Eq. (33b) in $O\left(2^{t-r}(t-r)t\right)$, we get the statement of Theorem 2.

The high level description of the proof of Lemma 4 goes as follows. First, we rewrite $V\left|0\right\rangle\!\left\langle 0\right|_{\text{abc}}^{\otimes n+t} V^\dagger$ appearing on the left hand side of Eq. (31a) as a stabilizer projector $\Pi_{\langle G^{(0)} \rangle}$ in the form of Eq. (30b). Viewing $\Pi_{\langle G^{(0)} \rangle}$ as a sum over stabilizers, we note that to contribute non-trivially to the sum in Eq. (31a), a stabilizer must satisfy certain constraints. In particular, for a fixed $g \in \langle G^{(0)} \rangle$ to produce a non-zero contribution to the sum, it is necessary that:

- Register 'a' constraints: for all $j \in [w]$, $|g|_j \in \{I, Z\}$,
- Register 'b' constraints: for all $j \in [n-w]$, $|g|_{w+j} = I$.

The generating set $\tilde{G} \in \mathcal{G}(n+t, t-r+v)$ is defined (and computed from $G^{(0)}$) such that the stabilizer group $\langle \tilde{G} \rangle$ contains $g \in \langle G^{(0)} \rangle$ if and only if g satisfies all of these constraints. From this $(n+t)$-qubit stabilizer group, we compute a "compressed" generating set, G, of a t-qubit stabilizer group $\langle G \rangle$. The quantity r is indirectly defined by:

$$|G| = t - r. \tag{34}$$

Thus, the stabilizer projector Π_G projects onto a subspace of dimension 2^r, i.e. $\text{Tr}\left(\Pi_G\right) = 2^r$. In addition one may impose constraints on the qubits from register 'c', as detailed in App. G. Exploring impact of imposing these constraints on the run-time of Compute and RawEstim is a work in progress, however we expect substantial improvements in the performance of Compute in the case of low Clifford count random circuits.

The quantity v is implicitly defined by the equation $|\tilde{G}| = t - r + v$. Together with related objects, J and x', the quantity v is associated with the compression step, i.e., transforming $\tilde{G}$ into G. Here, each generator $g \in \tilde{G}$ is mapped to a Pauli $f_x(g) \in \mathcal{P}_t$ where $f_x(g) := \omega(g) \left\langle x\right| |g|_{\text{a}} \left|x\right\rangle |g|_{\text{c}}$. The set $\left\{ f_x(g) | g \in \langle \tilde{G} \rangle \right\}$ is a group but the set of Pauli operators $\left\{ f_x(g) | g \in \tilde{G} \right\}$ may not be independent. That is, for a fixed $x \in \{0,1\}^w$ and $g^* \neq I^{\otimes n+t}$, it is possible that $f_x(g^*) \in \{\pm I^{\otimes t}\}$. When $f_x(g^*) = -I^{\otimes t}$, the sum over $g \in \tilde{G}$ of $f_x(g)$ is zero. The objects J and x' specify the constraints on x that ensure $-I^{\otimes t} \notin \left\{ f_x(g) | g \in \langle \tilde{G} \rangle \right\}$. When $f_x(g^*) = I^{\otimes t}$, the sum over $g \in \langle \tilde{G} \rangle$ of $f_x(g)$ contains duplicate sums over the group. The quantity v is the minimal number of deletions to $\tilde{G}$ required to ensure the image under f_x is an independent set.

C. Step 3: Gate sequence construction

So far we have replaced a general circuit U with a post-selected Clifford circuit V in Step 1, and then employed the stabilizer formalism in Step 2 to re-express the Born probability p using a compressed stabiliser projector Π_G. Now, the final step is to go back from the compressed projector picture to a compressed unitary circuit W built of Clifford gates. The aim of this step is summarised by the following lemma.

Lemma 5 (GateSeq subroutine)**.** *Given a stabilizer generator matrix $G \in \mathcal{G}(t, t-r)$,* GateSeq *outputs an elementary description of a t-qubit Clifford unitary W such that:*

$$\Pi_G = W^\dagger (\left|0\right\rangle\!\left\langle 0\right|^{\otimes t-r} \otimes I^{\otimes r}) W. \tag{35}$$

The circuit W consists of $O\left(t^2\right)$ Clifford gates including at most $O\left(t\right)$ Hadamard gates, and the run-time scaling of the algorithm is given by

$$\tau_{\text{GateSeq}} = \text{poly}(n, c, t). \tag{36}$$

The proof of the above lemma can be found in Appendix B, and it is simply based on an explicit construction of a circuit W out of elementary Clifford gates using the stabilizer formalism. Now, applying Lemma 5 to Eq. (33b) we immediately get

$$p = 2^{t-r+v-w} \left\| \left\langle 0\right|^{\otimes t-r} W \left|T_{\boldsymbol{\phi}}^\dagger\right\rangle \right\|_2^2, \tag{37}$$

which is precisely the main statement of Theorem 1 (with the second equality already proven in Eq. (33b)). Moreover, since Steps 1 to 3 all required polynomial number of operations, the total run-time of the Compress algorithm is $\text{poly}(n, c, t)$, and so we have proven Theorem 1.

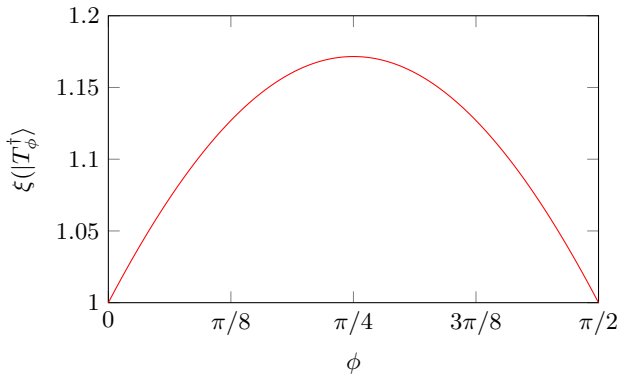


FIG. 6. **Stabiliser extent.** The values of the stabiliser extent ξ of $|T_\phi^\dagger\rangle$ states as a function of ϕ. Note that the maximum at $\phi = \pi/4$ is achieved for 2^γ with $\gamma \approx 0.228$ being exactly the same as in the exponential component of the run-time of the sampling algorithm presented in Ref. [29].

IV. THE RAWESTIM ALGORITHM

A. Step 1: Stabiliser decomposition and sampling

Each state $|T_{\phi_j}^\dagger\rangle$ appearing in $|T_{\boldsymbol{\phi}}^\dagger\rangle$ can be decomposed into stabilizer states,

$$\left|\tilde{0}\right\rangle := |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \qquad \left|\tilde{1}\right\rangle := |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle), \tag{38}$$

as follows:

$$|T_{\phi_j}^\dagger\rangle = \alpha_{\phi_j}\left|\tilde{0}\right\rangle + \alpha'_{\phi_j}\left|\tilde{1}\right\rangle, \tag{39}$$

where

$$\alpha_{\phi_j} = \frac{i + e^{-i\phi_j}}{1+i} = e^{i\varphi_j}\sqrt{1-\sin\phi_j}, \qquad \alpha'_{\phi_j} = \frac{1 - e^{-i\phi_j}}{1+i} = e^{i\varphi'_j}\sqrt{1-\cos\phi_j}, \tag{40}$$

for some phases φ_j, φ'_j.

The above decomposition achieves the minimum defining the stabiliser extent ξ [30],

$$\xi(|\psi\rangle) := \min_c \left\{ \|c\|_1^2 \;\middle|\; |\psi\rangle = \sum_j c_j\,|\sigma_j\rangle\,, \quad |\sigma_j\rangle \text{ is a stabiliser state} \right\}, \tag{41}$$

i.e.,

$$\xi(|T_{\phi_j}^\dagger\rangle) = (|\alpha_{\phi_j}| + |\alpha'_{\phi_j}|)^2 = (\sqrt{1-\sin\phi_j} + \sqrt{1-\cos\phi_j})^2. \tag{42}$$

We choose this particular decomposition because it minimises the run-time of the algorithm: as we will shortly see, it scales in the square of the l_1-norm of the expansion coefficients. Moreover, as proven in Ref. [30], the stabilizer extent for products of single-qubit states is multiplicative. Thus, denoting by ξ^* the total stabiliser extent of all states coming from reverse gadgetisation of non-Clifford gates in U, we have

$$\xi^* := \xi(|T_{\boldsymbol{\phi}}^\dagger\rangle) = \prod_{j=1}^{t} \xi(|T_{\phi_j}^\dagger\rangle), \tag{43}$$

and so the optimal stabiliser decomposition of $|T_{\boldsymbol{\phi}}^\dagger\rangle$ is simply obtained by decomposing each $|T_{\phi_i}^\dagger\rangle$ according to Eq. (39). In Fig. 6, we present the values of the stabiliser extent of $|T_\phi^\dagger\rangle$ as a function of ϕ.

Using the optimal stabiliser decomposition, we can rewrite Eq. (8) as follows

$$p = 2^{t-r+v-w} \left\| \sum_y \prod_{j=1}^{t} \alpha_{\phi_j}^{1-y_j} \alpha_{\phi_j}'^{y_j} \langle 0|^{\otimes t-r} W |\tilde{y}\rangle \right\|_2^2 \tag{44}$$

$$= \xi^* \cdot 2^{t-r+v-w} \left\| \sum_y \prod_{j=1}^{t} \frac{\alpha_{\phi_j}^{1-y_j} \alpha_{\phi_j}'^{y_j}}{|\alpha_{\phi_j}| + |\alpha_{\phi_j}'|} \langle 0|^{\otimes t-r} W |\tilde{y}\rangle \right\|_2^2 \tag{45}$$

$$= \xi^* \cdot 2^{t-r+v-w} \left\| \sum_y q(y) \prod_{j=1}^{t} e^{i\varphi_j(1-y_j)} e^{i\varphi_j' y_j} \langle 0|^{\otimes t-r} W |\tilde{y}\rangle \right\|_2^2, \tag{46}$$

where $q(y)$ is a normalised product probability distribution,

$$q(y) = \prod_{j=1}^{t} q(y_j), \quad q(y_j) = \begin{cases} \frac{|\alpha_{\phi_j}|}{|\alpha_{\phi_j}|+|\alpha_{\phi_j}'|} & \text{for } y_j = 0, \\ \frac{|\alpha_{\phi_j}'|}{|\alpha_{\phi_j}|+|\alpha_{\phi_j}'|} & \text{for } y_j = 1. \end{cases} \tag{47}$$

Therefore, we can introduce the following states:

$$|\psi(y)\rangle := \sqrt{\xi^*} \cdot 2^{\frac{t-r+v-w}{2}} \prod_{j=1}^{t} e^{i\varphi_j(1-y_j)} e^{i\varphi_j' y_j} \langle 0|^{\otimes t-r} W |\tilde{y}\rangle, \tag{48}$$

and write p as

$$p = \| |\mu\rangle \|_2^2, \qquad |\mu\rangle := \mathop{\mathbb{E}}_{Y\sim q} [|\psi(Y)\rangle] = \sum_y q(y) |\psi(y)\rangle. \tag{49}$$

We thus see that the Born rule probability p is given by the squared length of a vector $|\mu\rangle$ that is an expectation value over vectors $|\psi(y)\rangle$ distributed according to $q(y)$. The idea behind our algorithm is then to estimate this expectation value $|\mu\rangle$ using a mean $|\overline{\psi}\rangle$ over s samples:

$$|\overline{\psi}\rangle = \frac{1}{s} \sum_{j=1}^{s} |\psi_j\rangle, \tag{50}$$

where each $|\psi_j\rangle$ takes the value $|\psi(y)\rangle$ with probability $q(y)$. More precisely, in order to obtain each sample we first generate a t-bit string y bit by bit according to $q(y_j)$. This way we generate the state $|\tilde{y}\rangle$ with probability $q(y)$. We then evolve it by a Clifford W and project on $|0\rangle^{\otimes t-r}$ to finally obtain $|\psi(y)\rangle$ with probability $q(y)$. The evolution and projection can be performed efficiently and we describe how to do it in the next step. Here, assuming that we have s such samples, we bound the estimation error.

First, we note that by construction $|\overline{\psi}\rangle$ is an unbiased estimator of $|\mu\rangle$. Next, we use the following lemma, the proof of which can be found in Appendix C, to upper-bound the norm of each $|\psi(y)\rangle$.

Lemma 6 (Upper-bound for $\| |\psi(y)\rangle \|_2^2$)**.** *For every elementary description of p, the corresponding vectors $|\psi(y)\rangle$ defined in Eq. (48) are unnormalised stabilizer states with the squared l_2-norm upper-bounded by the total stabiliser extent ξ^* of all states coming from reverse gadgetisation of non-Clifford gates appearing in that elementary description:*

$$\| |\psi(y)\rangle \|_2^2 \leq \xi^*. \tag{51}$$

It is very important to note that the above bound for $\| |\psi(y)\rangle \|_2^2$ is general, i.e. independent of the particularities of a given quantum circuit. We do expect that stronger circuit-specific bounds can be efficiently computed, which would translate into improved run-times of the RawEstim algorithm. Now, the key technical tool that we will employ is the next lemma, proven in Appendix D, which applies a concentration inequality for vector martingales given by Heyes [40] to our setting.

Lemma 7. *Let $N, s \in \mathbb{N}$ and $\{|\psi_j\rangle\}_{j\in[N]}$ be a set of d-dimensional vectors over $\mathbb{C}$ satisfying $\||\psi_j\rangle\|_2^2 \leq m$. Moreover, let q be a probability distribution over $[N]$ and define $|\mu\rangle$ as the d-dimensional vector over $\mathbb{C}$ that is the expectation of $|\psi_X\rangle$ with respect to the random variable X with probability distribution q:*

$$|\mu\rangle = \mathop{\mathbb{E}}_{X\sim q}\left[|\psi_X\rangle\right] = \sum_{j\in[N]} q_j\,|\psi_j\rangle\,.$$

For $j \in [s]$, let $x_j \in [N]$ be independently sampled from the probability distribution q, and define a vector sample mean over s samples by:

$$\left|\overline{\psi}\right\rangle = \frac{1}{s}\sum_{j=1}^{s}\left|\psi_{x_j}\right\rangle. \tag{52}$$

Then, for all $\epsilon > 0$:

$$\Pr\left(\left\|\left|\overline{\psi}\right\rangle - |\mu\rangle\right\|_2 \geq \epsilon\right) \leq 2e^2\exp\left(\frac{-s\epsilon^2}{2(\sqrt{m}+\sqrt{p})^2}\right) \tag{53}$$

and

$$\Pr\left(\left\|\left|\overline{\psi}\right\rangle\!\left\langle\overline{\psi}\right| - |\mu\rangle\langle\mu|\right\|_1 \geq \epsilon\right) \leq 2e^2\exp\left(\frac{-s(\sqrt{p+\epsilon}-\sqrt{p})^2}{2(\sqrt{m}+\sqrt{p})^2}\right) \tag{54}$$

where $p := \||\mu\rangle\|_2^2$ and $\|\cdot\|_1$ is the Schatten 1-norm.

The bound on the estimation error, leading to the exponential scaling of the run-time (measured by the number of steps s) with the total stabiliser extent, can now be given as a simple corollary of the above technical lemmas.

Corollary 8 (Upper-bound for estimation error)**.** *The mean vector $\left|\overline{\psi}\right\rangle$ from Eq. (50) satisfies*

$$\Pr\left(\left|\left\|\left|\overline{\psi}\right\rangle\right\|_2^2 - p\right| \geq \epsilon\right) \leq \delta, \quad \delta := 2e^2\exp\left(\frac{-s(\sqrt{p+\epsilon}-\sqrt{p})^2}{2(\sqrt{\xi^*}+\sqrt{p})^2}\right). \tag{55}$$

Proof. We first note that $|\mathrm{Tr}\,(A)| \leq \|A\|_1$ for any Hermitian operator A. This follows from the fact that $\mathrm{Tr}\,(A)$ is the sum of the eigenvalues of A while $\|A\|_1$ is the sum of the singular values of A.By applying this inequality to $A = \left|\overline{\psi}\right\rangle\!\left\langle\overline{\psi}\right| - |\mu\rangle\langle\mu|$, the result follows immediately from Eq. (54), where m can be replaced by ξ^* due to Lemma 6. □

B. Step 2: State evolution

In the previous step we showed that by randomly sampling s stabiliser states $|\psi_j\rangle$, each equal to $|\psi(y)\rangle$ with probability $q(y)$, and creating their uniform superposition $\left|\overline{\psi}\right\rangle$, we can estimate p by calculating $\left\|\left|\overline{\psi}\right\rangle\right\|_2^2$. Here, we will show how to efficiently obtain the description of each sampled state. It is clear from Eq. (48) that to find a given $|\psi(y)\rangle$ it is enough to find an efficient way of representing $\langle 0|^{\otimes t-r} W\,|\tilde{y}\rangle$ for every y. This step of the algorithm will consist of three parts: first, we will explain how to get $W\left|\tilde{0}\ldots\tilde{0}\right\rangle$; then, how to modify this state to obtain $W\,|\tilde{y}\rangle$ for arbitrary y; and finally, how to perform post-selection to end up with $\langle 0|^{\otimes t-r} W\,|\tilde{y}\rangle$.

In the first part, we use the phase-sensitive Clifford simulator described in Ref. [30] to efficiently calculate the CH form of a t-qubit stabiliser state $W\left|\tilde{0}\ldots\tilde{0}\right\rangle$. The CH form of a general t-qubit stablizer state $|\sigma\rangle$ can be described by a tuple $\mathcal{T}(\sigma) = \{F, G, M, \gamma, v, s, \omega\}$. Here F, G and M are $t \times t$ binary matrices, γ is a length t vector with entries in $\mathbb{Z}_4$, v and s are binary vectors of length t, and ω is a complex number. Ref. [30] shows that for each gate $\Gamma \in \{S, CX, CZ\}$, the updated information $\mathcal{T}(\sigma')$ representing $|\sigma'\rangle = \Gamma\,|\sigma\rangle$ can be computed in $O\,(t)$ elementary steps. Updates associated with each Hadamard gate can be computed in $O\left(t^2\right)$ steps. Since W is composed of $O\left(t^2\right)$ including $O\,(t)$ Hadamard gates, in $O\left(t^3\right)$ steps we can calculate the CH form of $W\left|\tilde{0}\ldots\tilde{0}\right\rangle$ by updating the CH form of $\left|\tilde{0}\ldots\tilde{0}\right\rangle$ step by step with every application of the elementary Clifford gates composing W. For completeness, we provide a more detailed introduction of the CH form in Appendix E. Importantly, this first step can be performed as pre-computation, before any sampling of y is started.

In the second part of this step, we show how to update the CH form of $W\left|\tilde{0}\ldots\tilde{0}\right\rangle$ to get the CH form of $W\,|\tilde{y}\rangle$ after sampling a given y. If the k^{th} bit of a bitstring z is zero, and y is the same bitstring with the k^{th} bit set to one

then $|\tilde{y}\rangle = S_k^3 |\tilde{z}\rangle$. Hence $W|\tilde{y}\rangle = WS_k^3W^\dagger W|\tilde{z}\rangle$. In order to update the state $W|\tilde{0}\dots\tilde{0}\rangle$ to $W|\tilde{y}\rangle$, for arbitrary y we therefore pre-compute the t Clifford operators $WS_k^3W^\dagger$. By writing $S_k^3 = \frac{1}{\sqrt{2}}e^{-i\frac{\pi}{4}}(I + iZ_k)$ we can apply Lemma 4 of Ref. [30] to update a single bit of y in time $O(t^2)$. Transforming $W|\tilde{0}\dots\tilde{0}\rangle$ into $W|\tilde{y}\rangle$ for arbitrary y therefore takes time $O(t^3)$.

In the third and final part, we need to transform the CH form of a t-qubit stabilizer state $W|\tilde{y}\rangle$ into an r-qubit stabiliser state $\langle 0|^{\otimes t-r} W|\tilde{y}\rangle$. The authors of Ref. [30] explained how, in $O(t^2)$ steps, one can update the CH form of a given t-qubit state to simulate the action of a projector $|0\rangle\langle 0|^{\otimes t-r}$. Surprisingly, despite the fact that the resulting state is a product state, it is highly non-trivial to discard the $(t-r)$ measured qubits in the CH form description. This complication is related to the fact that the tuple $T(\sigma)$ corresponding to a stabiliser state $|\sigma\rangle$ is not unique. Thus, there exists a large number of equivalent tuples describing a given product state that do not admit a decomposition into two tuples representing each component of the tensor product. Therefore, we have developed a new subroutine allowing us to deal with this issue, and it is summarised in the following theorem, the proof of which can be found in Appendix F.

Lemma 9 (Discarding systems in CH form)**.** *Given a tuple describing the CH form of an $(n+1)$-qubit stabilizer state $|0\rangle \otimes |\sigma\rangle$, one can find the tuple describing the CH form of an n-qubit stabilizer state $|\sigma\rangle$ in $O(n^2)$ time.*

Using the above and given y, we can generate the CH form of a state $|\psi(y)\rangle$ in $O(t^2(t-r))$ time.

C. Step 3: Norm estimation

We are now at the point that we have the description of a state $|\overline{\psi}\rangle$ as a uniform superposition of s stabiliser states $|\psi_1\rangle, \dots |\psi_s\rangle$, and the squared l_2-norm of $|\overline{\psi}\rangle$ is an estimate for the Born rule probability p. The goal of the final step is to find and estimate $\hat{p}$ for $\||\overline{\psi}\rangle\|_2^2$ (so effectively for p), and bound the total estimation error by relating it to the run-time.

Given a vector $|\overline{\psi}\rangle$ with a decomposition into an s-term linear combination of r-qubit stabilizer states $|\psi_j\rangle$ from Eq. (50), one can estimate the squared l_2-norm of $|\overline{\psi}\rangle$ using a fast norm estimation algorithm by Bravyi and Gosset [29]. As inputs, the algorithm is given the CH-forms of $|\psi_j\rangle$. Next, it generates L randomly sampled r-qubit stabilizer states $|\theta_1\rangle, \dots, |\theta_L\rangle$. The estimate $\hat{p}$ is then given by:

$$\hat{p} = \frac{2^r}{s^2 L} \sum_{j=1}^{L} \left| \sum_{k=1}^{s} \langle \theta_j | \psi_k \rangle \right|^2 . \tag{56}$$

Each phase sensitive stabilizer inner product, $\langle \theta_j | \psi_k \rangle$, appearing above takes $O(r^3)$ steps to evaluate, and so we need $O(sLr^3)$ steps to evaluate $\hat{p}$. By choosing:

$$L = \left\lceil \tilde{\epsilon}^{-2} \log \tilde{\delta}^{-1} \right\rceil , \tag{57}$$

we ensure that the estimate $\hat{p}$ has multiplicative precision, i.e., for any desired error level, $\tilde{\epsilon} > 0$ and failure probability, $\tilde{\delta} > 0$, we have

$$\Pr\left(\left| \hat{p} - \||\overline{\psi}\rangle\|_2^2 \right| \geq \tilde{\epsilon} \||\overline{\psi}\rangle\|_2^2 \right) \leq \tilde{\delta}, \tag{58}$$

with the run-time scaling as:

$$O\left(sr^3 \ \tilde{\epsilon}^{-2} \log \tilde{\delta}^{-1} \right). \tag{59}$$

We have now estimated the squared l_2-norm of $|\mu\rangle$, i.e. p, by an estimate of the squared l_2-norm of $|\overline{\psi}\rangle$, i.e. $\hat{p}$. This introduced two sources of error. The first is due to the deviation between the two squared l_2-norms, and we have bounded this error in Corollary 8. The second source of error is due to the deviation between the estimate $\hat{p}$ and the squared l_2-norm of $|\overline{\psi}\rangle$. We now combine these two errors to show that, for an appropriate choice of s and L, our estimate satisfies Eq. (14). First, we can employ the triangle inequality to obtain

$$|\hat{p} - p| = \left| \hat{p} - \||\overline{\psi}\rangle\|_2^2 + \||\overline{\psi}\rangle\|_2^2 - p \right| \leq \left| \hat{p} - \||\overline{\psi}\rangle\|_2^2 \right| + \left| \||\overline{\psi}\rangle\|_2^2 - p \right| . \tag{60}$$

From Eq. (58) we have that with probability larger than $1-\tilde{\delta}$ the following holds:

$$\left|\hat{p}-\left\|\left|\overline{\psi}\right\rangle\right\|_2^2\right| \leq \tilde{\epsilon}\left\|\left|\overline{\psi}\right\rangle\right\|_2^2 \leq \tilde{\epsilon}\left(\left|\left\|\left|\overline{\psi}\right\rangle\right\|_2^2-p\right|+p\right)=\tilde{\epsilon}\left(\epsilon+p\right). \tag{61}$$

Then, from Eq. (55) we get that with probability larger than $1-\delta$ we have

$$\left|\left\|\left|\overline{\psi}\right\rangle\right\|_2^2-p\right| \leq \epsilon. \tag{62}$$

Since both steps (computing the sample average vector $\left|\overline{\psi}\right\rangle$ and computing $\hat{p}$ using fast norm estimation) are independent, we get that with probability larger than $(1-\delta)(1-\tilde{\delta})$ we have

$$|\hat{p}-p| \leq \tilde{\epsilon}(\epsilon+p)+\epsilon. \tag{63}$$

We can thus write

$$\Pr\left(|\hat{p}-p| \geq \tilde{\epsilon}(\epsilon+p)+\epsilon\right) \leq \tilde{\delta}+\delta. \tag{64}$$

Introducing variables describing the total estimation error, $\epsilon_{\text{tot}}>0$ and $\delta_{\text{tot}}>0$, we want to find the bounds on the number of samples s from Step 1 and on the number of repetitions L from Step 3, so that the estimate $\hat{p}$ satisfies:

$$\Pr\left(|\hat{p}-p| \geq \epsilon_{\text{tot}}\right) \leq \delta_{\text{tot}}. \tag{65}$$

Employing Eq. (64), together with Eqs. (55) and (57), the above is satisfied whenever for any arbitrary choice of $\epsilon \in (0,\epsilon_{\text{tot}})$ and $\delta \in (0,\delta_{\text{tot}})$ we have

$$s \geq \frac{2(\sqrt{\xi^*}+\sqrt{p})^2}{\left(\sqrt{p+\epsilon}-\sqrt{p}\right)^2}\log\left(\frac{2e^2}{\delta}\right), \tag{66a}$$

$$L \geq \left(\frac{p+\epsilon}{\epsilon_{\text{tot}}-\epsilon}\right)^2\log\left(\frac{1}{\delta_{\text{tot}}-\delta}\right). \tag{66b}$$

The output of the fast norm estimation algorithm $\hat{p}$ is the output of our RawEstim algorithm. The bound on the estimation error, Eq. (65), together with the bounds on s and L, Eqs. (66a) and (66b), are equivalent to the main statement of Theorem 3. To finish the proof, we need to show that the run-time is indeed as claimed in the theorem. To see this, recall that producing each of s samples $|\psi_j\rangle$ (sampling y in Step 1 and evolving the state in Step 2) takes $O\left(t^2(t-r)\right)$. Moreover, each sample has to go through L repetitions of the norm estimation subroutine, with each repetition taking $O\left(r^3\right)$ steps. Putting this all together, the run-time of the RawEstim algorithm is $O\left(st^2(t-r)+sLr^3\right)$, as claimed in Theorem 3.

We present the psuedocode for the RawEstim algorithm below.

Algorithm 1 RawEstim outputs an estimate $\hat{p}$ as characterized by Theorem 3.

Input: Output of Compress for an elementary description of p and parameters $s, L \in \mathbb{N}$.
Output: An estimate $\hat{p}$.

```
[r, v, W] ← Compress(D)                    ▷ D represents the elementary description of p.
Compute CH-Form(r, W, 0̃)                   ▷ This is the CH-form of the state ⟨0|^{⊗t−r} W |0̃⟩^{⊗t}.
for k ∈ [s] do
    y ← Y where Y ∈ {0,1}^t is sampled according to the product distribution in Eq. (47)
    Compute |ψ_k⟩ ← CH-Form(r, W, ỹ)       ▷ CH-Form(r, W, ỹ) is computed from CH-Form(r, W, 0̃) as per Sec. IV B.
end for
for j ∈ [L] do
    |θ_j⟩ ← random r-qubit equatorial state.
end for
p̂ ← SumOverlaps({|ψ_k⟩}_{k∈[s]}, {|θ_j⟩}_{j∈[L]})   ▷ The SumOverlaps sub-procedure evaluates Eq. (56).
return p̂
```

Finally, we note that both s and L depend on the unknown quantity p and increase with p. Thus if we require a bound on our estimate's failure probability, we can make a conservative choice by substituting $p=1$ into Eqs. (66a) and (66b). Instead of this naive approach, the Estimate algorithm allows us to significantly improve run-time by making a less conservative choice of p thus taking advantage of the substantial run-time improvements that occur for smaller p values.

V. THE ESTIMATE ALGORITHM

The role of the ESTIMATE algorithm is to choose the parameters used in making repeated calls to the RAWESTIM algorithm with the goal of finally attaining a Born rule probability estimate $\hat{p}$ satisfying Eq. (16). The goal is to achieve this task with frugal use of run-time. Consistent with Eq. (15), we model the run-time of RAWESTIM using Eq. (17). Hence the ESTIMATE algorithm aims to minimize the run-time cost, $\mathcal{C}$, as defined in Eq. (18).

For $p \in [0,1]$, $\epsilon_{\text{tot}} \in \mathbb{R}^+$, $\eta \in (0,1)$ and $s, L \in \mathbb{N}^+$, we define the function:

$$\delta'(p, \epsilon_{\text{tot}}, \eta, s, L) := 2e^2 \exp\left(\frac{-s(\sqrt{p+\eta\epsilon_{\text{tot}}} - \sqrt{p})^2}{2(\sqrt{\xi^*}+1)^2}\right) + \exp\left(-\left(\frac{(1-\eta)\epsilon_{\text{tot}}}{p+\eta\epsilon_{\text{tot}}}\right)^2 L\right). \tag{67}$$

Comparing to Eq. (14), we note that RAWESTIM(s, L) outputs an estimate $\hat{p}$ such that for all $\epsilon_{\text{tot}} > 0$ and $\eta \in (0,1)$:

$$\Pr\left(|\hat{p} - p| \geq \epsilon_{\text{tot}}\right) \leq \delta'(p, \epsilon_{\text{tot}}, \eta, s, L). \tag{68}$$

For fixed p, η, s, L, we want to view δ' as a function of ϵ_{tot} and define its functional inverse. We will need this to be defined for all $\delta' > 0$. By inspection of Eq. (67), we note that for $\delta_{\text{targ}} > 0$ close to zero and L, η too small there does not exits ϵ' such that $\delta'(p, \epsilon', \eta, s, L, m) = \delta_{\text{targ}}$. To resolve this technicality, we define a minimal L value:

$$L_{\min}(\delta, \eta) := \left\lceil -\left(\frac{\eta}{(1-\eta)}\right)^2 \ln \delta \right\rceil. \tag{69}$$

To specify a well defined inverse $\epsilon'(p, \delta_{\text{targ}}, \eta, s, L) \in \mathbb{R}^+$ of the δ' function, let us define its domain

$$D = [0,1] \times (0,1) \times (0,1) \times \mathbb{N}^+ \times \mathbb{N}^+. \tag{70}$$

By inspecting Eq. (67), it is clear that on D, there exists a well defined function ϵ' that satisfies the following: for all $(p, \delta_{\text{targ}}, \eta, s, L^+) \in D$, there exists $\epsilon_{\text{targ}} =: \epsilon'(p, \delta_{\text{targ}}, \eta, s, L^+)$ such that $\delta'(p, \epsilon_{\text{targ}}, \eta, s, L_{\min}(\delta_{\text{targ}}, \eta) + L^+) = \delta_{\text{targ}}$. To see this, we just note that for $p \in [0,1]$ fixed, the function $f(c) = \sqrt{p+c} - \sqrt{p}$ is strictly increasing and unbounded from above.

A property of $\epsilon'(p, \delta_{\text{targ}}, \eta, s, L^+)$ that will be useful is that it is a monotonically increasing function of p. To see this we note that by the definition of the functions ϵ' and δ', we have:

$$0 = d\delta'(p, \epsilon', \eta, s, L_{\min}(\delta, \eta)) = \frac{\partial \delta'}{\partial p} dp + \frac{\partial \delta'}{\partial \epsilon'} d\epsilon'. \tag{71}$$

Using Eq. (67), it is easy to verify that $\frac{\partial \delta'}{\partial p} \geq 0$ and $\frac{\partial \delta'}{\partial \epsilon'} \leq 0$. Thus $\frac{d\epsilon'}{dp} \geq 0$.

For $\mathcal{T} \in [2, \infty)$ and all other ranges as before, let us define the function $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T}) \in \mathbb{R}$ as:

$$\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T}) = \inf_{\eta, s, L^+} \epsilon'(p, \delta_{\text{tot}}, \eta, s, L^+) \tag{72}$$

where the infimum is over all $\eta \in (0,1)$, $s, L^+ \in \mathbb{N}^+$ subject to the constraint:

$$\tau_{\text{model}}(s, L^+ + L_{\min}(\delta_{\text{tot}}, \eta)) \leq \mathcal{T}. \tag{73}$$

Since the range $\eta \in (0,1)$ is not closed, in principal the function ϵ' could get arbitrarily close to its infimum ϵ^* without attaining it. However, one can show that for all $p \in [0,1]$, $\delta_{\text{tot}} \in (0,1)$ and $\mathcal{T} \geq 2$, there always exists $\eta \in (0,1)$ and $s, L^+ \in \mathbb{N}^+$ such that the infimum is attained, i.e. $\epsilon'(p, \delta_{\text{tot}}, \eta, s, L^+) = \epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})$. To see this, we note that Eqs. (69) and (73) can be used to impose a closed upper bound on η. Similarly, using the fact that in the limit of $\eta \to 0$, $\delta'(p, \epsilon_{\text{tot}}, \eta, s, L)$ becomes greater than 1, we can impose a closed lower bound on η. Having restricted the range of η in Eq. (72) to a closed interval contained in $(0,1)$, we can apply the Extreme Value Theorem to prove our claim.

We now show that for $\delta_{\text{tot}} \in (0,1]$ and $\mathcal{T} \geq 2$ fixed, $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})$ is monotonically increasing in p. Let η', s', L' and η'', s'', L'' be such that $\epsilon^*(p', \delta_{\text{tot}}, \mathcal{T}) = \epsilon'(p', \delta_{\text{tot}}, \eta', s', L')$ and $\epsilon^*(p'', \delta_{\text{tot}}, \mathcal{T}) = \epsilon'(p'', \delta_{\text{tot}}, \eta'', s'', L'')$. Then, for $p' \leq p''$, we have:

$$\begin{aligned}
\epsilon^*(p', \delta_{\text{tot}}, \mathcal{T}) &= \epsilon'(p', \delta_{\text{tot}}, \eta', s', L') \\
&\leq \epsilon'(p', \delta_{\text{tot}}, \eta'', s'', L'') \\
&\leq \epsilon'(p'', \delta_{\text{tot}}, \eta'', s'', L'') \\
&= \epsilon^*(p'', \delta_{\text{tot}}, \mathcal{T}),
\end{aligned} \tag{74}$$

where the first inequality holds by the definition of ϵ^* and the second inequality holds by the fact that ϵ' is monotone increasing in p.

The ESTIMATE algorithm works by iteratively querying the RAWESTIM algorithm with each iteration indexed by $k = 1, 2, \ldots$. Each call RAWESTIM(s_k, L_k) allocates a run-time budget $\mathcal{T}_k$ to RAWESTIM so that $\tau_{\text{model}}(s_k, L_k) \leq \mathcal{T}_k$. The budget allocation in the first step is $2\mathcal{T}_0$ where $\mathcal{T}_0$ is a budget that is insufficient (in the best case scenario of $p = 0$) to satisfy Eq. (16). Starting from this low initial run-time allocation, the budget doubles at each iteration. Thus, the run-time budget for each call of the RAWESTIM algorithm and the total run-time over all prior calls both grow exponentially in the round number.

Each round's estimate $\hat{p}_k$ is used to compute a probabilistic quantity p^*_k, that with high probability upper bounds p. From Eq. (14) and the definitions of ϵ' and ϵ^*, we note that for all $\delta_{\text{tot}} > 0$ and η, s, L^+ satisfying $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T}) = \epsilon'(p, \delta_{\text{tot}}, \eta, s, L^+)$, the output $\hat{p}$ of RAWESTIM$(s, L_{\min}(\delta_{\text{tot}}, \eta) + L^+)$ satisfies:

$$\Pr\left(|\hat{p} - p| \geq \epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})\right) \leq \delta_{\text{tot}}. \tag{75}$$

Thus, defining $p^*_{\min}$ as the random variable constructed from $\hat{p}$ using $p^*_{\min} = \hat{p} + \epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})$ we have:

$$\Pr\left(p^*_{\min} \leq p\right) \leq \delta_{\text{tot}}, \tag{76}$$

where the probability is over the randomness of the RAWESTIM algorithm. Of course, given a $\hat{p}$ one cannot compute the corresponding upper bound $p^*_{\min}$ because this requires the evaluation of $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})$ with p unknown. We overcome this by using the fact that $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T})$ is monotonically increasing in p. Thus, if p^*_k is a probabilistic upper bound of p with failure probability $\delta_1 + \ldots + \delta_k$, then, by the union bound, $p^*_{k+1} := \hat{p} + \epsilon^*(p^*_k, \delta_{k+1}, \mathcal{T})$ is a probabilistic upper bound for p with failure probability $\delta_1 + \ldots + \delta_{k+1}$. Using this procedure, we can start with the upper bound of 1 and iteratively produce tighter upper bounds at the cost of incurring a increased probability of failure of the upper bound. We will choose $\delta_k := \frac{6}{\pi^2 k^2}\delta_{\text{tot}}$. This guarantees that the infinite sum $\delta_1 + \delta_2 + \ldots$ converges to δ_{tot} and hence the probability that at least one of the upper bounds, $p^*_1, p^*_2 \ldots$, fails is at most δ_{tot}.

For $(p, \delta_{\text{tot}}, \mathcal{T}) \in [0, 1] \times \mathbb{R}^+ \times [2, \infty)$, we define the function $OptC(p, \delta_{\text{tot}}, \mathcal{T}) \in (0, 1) \times \mathbb{N}^+ \times \mathbb{N}^+$. For a given target cost $\mathcal{T}$, this function finds the optimal choices η, s and L^+ (subject to the cost budget constraint) such that $\epsilon^*(p, \delta_{\text{tot}}, \mathcal{T}) = \epsilon'(p, \delta_{\text{tot}}, \eta, s, L^+)$.

We now present the pseudocode for the ESTIMATE algorithm before discussing the characterization of the run-time of ESTIMATE.

Algorithm 2 ESTIMATE returns the estimate $\hat{p}$ as characterized by Eq. (16).

Input: Output of COMPRESS for an elementary description of p and accuracy parameters $\epsilon_{\text{tot}}, \delta_{\text{tot}} > 0$.
Output: An estimate $\hat{p}$.

$[r, v, W] \leftarrow$ COMPRESS$(\mathcal{D})$ ▷ $\mathcal{D}$ represents the elementary description of p.
$p^* \leftarrow 1$ ▷ This is the running upper bound for the unknown p.
Exit $\leftarrow 0$, $k \leftarrow 0$
$\mathcal{T}_0 \leftarrow \tau_{\text{model}}(-\frac{2(\sqrt{\xi^*}+1)^2}{\epsilon_{\text{tot}}} \ln \frac{\delta_{\text{tot}}}{2e^2}, 1)$ ▷ $\mathcal{T}_0$ is a runtime budget that is too small to satisfy Eq. (16) even assuming $p = 0$.
while Exit $= 0$ **do**
  $k \leftarrow k + 1$
  $(\eta, s, L^+) \leftarrow OptC(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k \mathcal{T}_0)$ ▷ In each round, we double the run-time budget.
  $\epsilon^* \leftarrow \epsilon'(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, \eta, s, L^+)$ ▷ Equivalently, $\epsilon^* \leftarrow \epsilon^*(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k \mathcal{T}_0)$.
  if $\epsilon^* \leq \epsilon_{\text{tot}}$ **then**
    Exit $\leftarrow 1$
  end if
  $\hat{p} \leftarrow$ RAWESTIM$(s, L^+ + L_{\min}(\frac{6}{\pi^2 k^2}\delta_{\text{tot}}, \eta))$
  $p^* \leftarrow \max\{0, \min\{1, p^*, \hat{p} + \epsilon^*\}\}$
end while
return $\hat{p}$

We note that small improvements in performance can be achieved by using a larger choice of $\mathcal{T}_0$ subject to the requirement that $\mathcal{T}_0$ is still too small to satisfy Eq. (16) even assuming $p = 0$. Fig. 3 shows the output and intermediate values of $\hat{p}, p^*$ and ϵ^* produced using the ESTIMATE algorithm.

The remainder of this section focuses on computing an upper bound for the total modeled run-time cost associated with ESTIMATE as defined in eq. (18). The run-time cost of ESTIMATE is probabilistic and dependent on the unknown p. Here, we introduce our RUNTIME algorithm which produces run-time cost upper bounds for any given p value. This algorithm can be used to produce a run-time cost upper bound as a function of p. We point out that the actual

run-time of ESTIMATE may differ from the run-time cost for a number of reasons. Firstly, run-time cost is a modelled and/or expected run-time and may differ from actual run-time it aims to predict due to limitations of the model or incorrectly calibrated model parameters c_1 and c_2. Secondly, the run-time cost only aims to model the total run-time of RAWESTIM over all calls made in Step. 12 of the Alg. 2. Thus, it ignores the run-time incurred by ESTIMATE in all other steps. We justify the choice to not model the run-time cost associated with these other steps since their run-time is insensitive to circuit parameters.

We now present the pseudo-code for our RUNTIME algorithm. We note that all steps except Steps 12 and 15 are identical to the pseudo-code for ESTIMATE.

Algorithm 3 RUNTIME returns a probabilistic upper bound of $\mathcal{C}$, the run-time cost defined in Eq. (18).

Input: Assumed value of p; $\delta_{\text{UB}} > 0$, the required maximum failure probability of the probabilistic upper bound for $\mathcal{C}$; the output of COMPRESS; and accuracy parameters $\epsilon_{\text{tot}}, \delta_{\text{tot}} > 0$.
Output: The probabilistic upper bound $\mathcal{C}_{\text{UB}} = \mathcal{C}_{\text{UB}}(p)$.

1: $[r, v, W] \leftarrow$ COMPRESS($\mathcal{D}$) ▷ $\mathcal{D}$ represents the elementary description of p.
2: $p^* \leftarrow 1$ ▷ This is the running upper bound for the unknown p.
3: Exit $\leftarrow 0$, $k \leftarrow 0$
4: $\mathcal{T}_0 \leftarrow \tau_{\text{model}}(-\frac{2(\sqrt{\xi^*}+1)^2}{\epsilon_{\text{tot}}} \ln \frac{\delta_{\text{tot}}}{2e^2}, 1)$ ▷ $\mathcal{T}_0$ is a runtime budget that is too small to satisfy Eq. (16) even assuming $p = 0$.
5: **while** Exit $= 0$ **do**
6: $k \leftarrow k + 1$
7: $(\eta, s, L^+) \leftarrow OptC(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k \mathcal{T}_0)$ ▷ In each round, we double the run-time budget.
8: $\epsilon^* \leftarrow \epsilon'(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, \eta, s, L^+)$ ▷ Equivalently, $\epsilon^* \leftarrow \epsilon^*(p^*, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k \mathcal{T}_0)$.
9: **if** $\epsilon^* \leq \epsilon_{\text{tot}}$ **then**
10: Exit $\leftarrow 1$
11: **end if**
12: $\hat{p} \leftarrow p + \epsilon'\left(p, \delta_{\text{UB}}/K_{\text{UUB}}, \tilde{\eta}, s, L^+ + L_{\min}\left(\frac{6}{\pi^2 k^2}\delta_{\text{tot}}, \eta\right) - L_{\min}(\delta_{\text{UB}}/K_{\text{UUB}}, \tilde{\eta})\right)$ ▷ The choice of $K_{\text{UUB}} > 0$ and $\tilde{\eta} \in (0, 1)$ are discussed below.
13: $p^* \leftarrow \max\{0, \min\{1, p^*, \hat{p} + \epsilon^*\}\}$
14: **end while**
15: **return** $\mathcal{C}_{\text{UB}} \leftarrow 2^{k+1}\mathcal{T}_0$

To establish the correctness of our RUNTIME algorithm, we first establish some notation. The ESTIMATE algorithm generates the following strings of random variables: $\{\hat{p}_k\}_{k\in[K]}$, $\{\epsilon^*_k\}_{k\in[K]}$, $\{p^*_k\}_{k\in[K]}$ and the string of triples $\left\{(\eta_k, s_k, L^+_k)\right\}_{k\in[K]}$ where K is itself a random variable indicating when the exit condition is triggered. The exit condition is triggered when

$$\epsilon^*_k := \epsilon^*\left(p^*_{k-1}, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k\mathcal{T}_0\right) \leq \epsilon_{\text{tot}} \tag{77}$$

for the first time. The lowest value of k for which the exit condition is triggered defines the random variable K.

Since $\tau_{\text{model}}(s_k, L_k) \leq 2^k\mathcal{T}_0$, where $L_k = L^+_k + L_{\min}(\frac{6}{\pi^2 k^2}\delta_{\text{tot}}, \eta_k)$, the total cost associated with calls to the RAWESTIM algorithm as modelled by Eq. (18) is upper bounded by:

$$\mathcal{C} \leq (2 + 2^2 + \ldots + 2^K)\mathcal{T}_0 < 2^{K+1}\mathcal{T}_0. \tag{78}$$

We note that the run-time cost is a random variable that depends on K. We will show that K_{UB}, the value of k used in Step 15 of the RUNTIME pseudo-code, is a probabilistic upper bound for K and hence $\mathcal{C} \leq \mathcal{C}_{\text{UB}}$ with probability $\geq 1 - \delta_{\text{UB}}$.

We note that the RUNTIME algorithm is deterministic. In the ESTIMATE algorithm, the randomness of the variables K, ϵ^*_k, p^*_k is entirely due to their functional dependence on $\{\hat{p}_k\}_{k\in[K]}$. The stochastic $\hat{p}_k$ used in Step 12 of the ESTIMATE algorithm are replaced with deterministic $\hat{p}_k$ in Step 12 of the RUNTIME algorithm. Thus, the associated strings of variables generated by the RUNTIME algorithm are all deterministic.

Let $\boldsymbol{p} = \{p_k\}_{k\in\mathbb{N}^+}$ be a sequence of probabilities $p_k \in [0, 1]$. Then we will use $K(\boldsymbol{p})$ and $\{\epsilon^*{}_k(\boldsymbol{p})\}_{k\in[K(\boldsymbol{p})]}$ to denote the values computed by ESTIMATE in the setting when the RAWESTIM algorithm's output is forced to be exactly the sequence $\boldsymbol{p}$. Let $\boldsymbol{p} = \{p_k\}_{k\in\mathbb{N}^+}$ be some sequence of probabilities $p_k \in [0, 1]$ representing the output of RAWESTIM. For $k = 1, 2, \ldots$, the variable ϵ^*_k computed in Line 8 can be specified by the recursion:

$$\epsilon^*_k(\boldsymbol{p}) = \epsilon^*\left(\max\left\{0, \min\left\{1, p_0 + \epsilon^*_0(\boldsymbol{p}), \ldots, p_{k-1} + \epsilon^*_{k-1}(\boldsymbol{p})\right\}\right\}, \frac{6}{\pi^2 k^2}\delta_{\text{tot}}, 2^k\mathcal{T}_0\right), \tag{79}$$

where $\epsilon_1^*(\boldsymbol{p}) := \epsilon^*(1, 6\delta_{\rm tot}/\pi^2, 2\mathcal{T}_0)$. From Eq. (79) and that $\epsilon^*(p, \delta, \mathcal{T})$ is monotone increasing in p, it is clear that higher values of p_{k-1} and ϵ_{k-1}^* both result in higher values of ϵ_k^*. Thus, for some fixed $\boldsymbol{p}$ with $p_1, \ldots, p_{k-1}$ sufficiently large, the deterministic quantity $\epsilon_k^*(\boldsymbol{p})$ is a probabilistic upper bounded of the random variable $\epsilon_k^* = \epsilon_k^*(\hat{p}_1, \hat{p}_2, \ldots)$ computed in the ESTIMATE algorithm. In particular, let us define $\boldsymbol{p} = \{p_k\}_{k\in\mathbb{N}^+}$ as per Step 15 of the RUNTIME algorithm:

$$p_k := p + \epsilon' \left(p, \delta_{\rm UB}/K_{\rm UUB}, \tilde{\eta}_k, s_k, L_k^+ + L_{\min}\left(\frac{6}{\pi^2 k^2}\delta_{\rm tot}, \eta_k\right) - L_{\min}(\delta_{\rm UB}/K_{\rm UUB}, \tilde{\eta}_k)\right). \tag{80}$$

Here, η_k, s_k and L_k^+ are parameters computed on the $k^{\rm th}$ iteration of Step 7 of RUNTIME but $\tilde{\eta}_k$ is a new independent parameter. We will see that any choice of $\tilde{\eta}_k \in (0, 1)$ will result in the desired upper bound and hence we can optimize the choice of $\tilde{\eta}_k$ to achieve a tighter upper bound. Although $K_{\rm UUB}$ must be chosen before $K_{\rm UB}$ can be computed, $K_{\rm UUB}$ can be any quantity that satisfies $K_{\rm UUB} \geq K_{\rm UB}$. Due to the weak dependence of $K_{\rm UB}$ on $K_{\rm UUB}$, such a choice is always possible. We note that the deterministic quantity p_k serves as a probabilistic upper bound of $\hat{p}_k \leftarrow \text{RAWESTIM}(s_k, L_k^+ + L_{\min}(\frac{6}{\pi^2 k^2}\delta_{\rm tot}, \eta_k))$ such that the probability that p_k fails to upper bound $\hat{p}_k$ for any $k \in \mathbb{N}^+$ is $\leq \delta_{\rm UB}/K_{\rm UUB}$. Further, since Eq. (68) holds for all η, our statement holds for all choices of $\tilde{\eta}_k \in (0, 1)$ subject to $L_k^+ + L_{\min}\left(\frac{6}{\pi^2 k^2}\delta_{\rm tot}, \eta_k\right) - L_{\min}(\delta_{\rm UB}/K_{\rm UUB}, \tilde{\eta}_k) \geq 1$. Thus, $K > K_{\rm UB}$ implies that there is a $\kappa \in [K_{\rm UB}]$ that is the smallest $k \in [K_{\rm UB}]$, such that $\hat{p}_k$ produced in Step 12 of ESTIMATE exceeds $\hat{p}_k$ produced in Step 12 of RUNTIME. By the union bound and our choice of p_k the probability of this happening is $\leq K_{\rm UB}\delta_{\rm UB}/K_{\rm UUB}$.

This implies that:

$$\Pr\left(K \leq K_{\rm UB}\right) \geq 1 - \delta_{\rm UB}. \tag{81}$$

We note that before any costly calls to the RAWESTIM algorithm are made, $\mathcal{C}_{\rm UB}$ can easily by computed and plotted as a function of p thus predicting probabilistic run-time upper bounds conditional on the unknown p. A similar plot is presented in Fig. 4 for the probabilistic upper bound of run-time cost $\mathcal{C}$.

Finally, we note that since the functions $\epsilon'(p, \delta_{\rm tot}, \eta, s, L - L_{\min}(\delta_{\rm tot}, \eta))$ and $OptC(\mathcal{T}, p, \delta_{\rm tot})$ are not given in a closed form, their evaluation requires using numerical techniques. These will inevitably be subject to small levels of imprecision with run-times that mildly (logarithmically or poly-logarithmically) depend on precision requirements. In principle, the run-time for the numerical evaluation of these functions depends on the Born probability estimation problem parameters such as $\epsilon_{\rm tot}$ since the precision parameters must be $\ll \epsilon_{\rm tot}$. In practice, the precision parameters are so small that $\epsilon_{\rm tot}$ values of this order would produce completely infeasible run-times for the RAWESTIM algorithm. Thus, we ignore such run-time dependencies and treat the evaluation of these functions as having a fixed cost independent of the estimation problem parameters.

VI. CONCLUSIONS AND OUTLOOK

We have developed state-of-the-art classical simulators for computing and estimating Born rule probabilities associated with universal quantum circuits. We have made Python+C implementations of these simulators available [35]. These simulators allow us to probe the previously uncharted parameter regimes, for circuits with larger numbers of qubits and non-Clifford gates than was previously possible. Our results should find direct applications in the verification and validation of near-term quantum devices, and the evaluation of proposals for NISQ device applications.

Through the use of our COMPRESS algorithm we were able to distill a complex circuit specification to a simpler form more amenable to the task of Born rule probability estimation. The circuit specific parameter, r, emerged as a key driver of run-time, with higher values of r improving the run-time of COMPUTE and lower values of r (often) improving the run-time of ESTIMATE. Thus the projector rank r is useful in identifying which simulator will be the fastest. In our work, the primary role served by COMPUTE has been to exclude all of the 'high r value' circuits from consideration, thus emphasising the performance advantages of our ESTIMATE algorithm over its alternatives. However, it remains an open question if and when the COMPUTE algorithm can be useful in its own right or has a genuinely interesting application. Indeed, in the extreme regime where $r = t$, COMPUTE outputs a Born rule probability consistent with the uniform distribution on all measured 'non-deterministic' qubits. However, as r moves away from t, perhaps the quantity $t - r$ (or other information contained in the stabilizer generating set G) can be viewed as a measure of departure from 'non-uniformity'. This is broadly consistent with the outcome of our numerical analysis of high Clifford count randomly generated circuits where we found that r strongly concentrates near its maximum value $\min\{t, n - w\}$. We leave the exploration of this narrative and the identification of other key drivers of outcome distribution structure to future work.

This work has focused on the task of Born rule probability estimation without discussing the related task of approximately sampling from the quantum outcome distribution. Some of the techniques we have developed here

may also be useful for achieving performance improvements for the task of approximate sampling. We leave this to future work.

We have restricted our attention to the simulation of ideal or noise-free quantum processes. Realistic implementations of quantum circuits are subject to noise the presence of which can significantly ease the computational cost of classical simulation. We leave open the generalization of our work to the mixed state formalism. We point out that for the task of approximate sampling, an analogous generalization (to the BG-sampling algorithm [29]) was recently shown in Ref. [22] with additional performance gains being achieved as a consequence of this generalization.

ACKNOWLEDGEMENTS

HP acknowledges Marco Tomamichel for identifying an error in the statement of Lem. 7 in an early draft; David Gosset for useful discussions regarding the CH-form; and Daniel Grier and Luke Schaeffer for useful discussions regarding the hardness of computing tight upper bounds associated with Lem. 6. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. HP also acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). KK and ORS acknowledge financial support by the Foundation for Polish Science through TEAM-NET project (contract no. POIR.04.04.00-00-17C1/18-00). This work is supported by the Australian Research Council (ARC) via the Centre of Excellence in Engineered Quantum Systems (EQuS) project number CE170100009. Research was partially sponsored (SB) by the ARO and was accomplished under Grant Number: W911NF-21-1-0007. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARO or the U.S. Government.

Appendix A: Proof of Lemma 4

We define the following useful form for a generating set.

Definition 10. For $G = \{g_1, \ldots, g_k\} \in \mathcal{G}(n,k)$ and $j \in [n]$, we say that G is in $ZX(j)$-form iff $\text{Type}(|g_1|_j, \ldots, |g_k|_j)$ belong to $\{[k-2,1,0,1],[k-1,1,0,0],[k-1,0,1,0],[k-1,0,0,1],[k,0,0,0]\}$ where for $P_1, \ldots, P_k \in \{I,X,Y,Z\}$, $\text{Type}(P_1, \ldots, P_k) = [N_I, N_X, N_Y, N_Z]$ indicates that there are exactly N_I occurrences of I, N_X occurrences of X and so on in the list $P_1, \ldots, P_k$.

When G is in $ZX(j)$-form, we call any generator $g \in G$ a *leading* generator if $|g|_j \neq I$. We call g the *leading-X* generator if $|g|_j = X$ and similarly for Y and Z. Two generating sets G and G' are equivalent iff they generate the same stabilizer group i.e. $\langle G \rangle = \langle G' \rangle$. If $g_1, g_2 \in G$ are distinct then replacing g_2 by $g_1 g_2$ produces an equivalent generating set. By repeatedly using this method, any generating set G can be transformed to an equivalent generating set G' such that G' is in $ZX(j)$-form for any suitable choice of j.

We now present a Lemma that will be useful for the proof of Lemma 4.

Lemma 11. *Let $n \in \{2,3,\ldots\}$ and $k \in \{0,1,\ldots,n-2\}$. Let $G = \{g_z, g_x, g_1, \ldots, g_k\} \in \mathcal{G}(n,k+2)$ be a stabilizer generating set in $ZX(j)$-form such that it has both a leading-Z and a leading-X generator, g_z and g_x respectively. For $a \in \{0,1\}$ fixed, let $\tilde{g}_z := \langle a| g_z |a\rangle$ and for $i \in [k]$, $\tilde{g}_i := \langle a| g_i |a\rangle$ where the $|a\rangle$ vectors act on the j^{th} qubit. Then $\tilde{G} := \{\tilde{g}_z, \tilde{g}_1, \ldots, \tilde{g}_k\} \in \mathcal{G}(n-1,k+1)$.*

Proof. It is clear that $\tilde{G} \subset \mathcal{P}_{n-1}$ is commuting so we only need to show that the subset is independent. The independence of the set $\{\tilde{g}_1, \ldots, \tilde{g}_k\}$ is inherited from the independence of G. Hence we only need to show that $\tilde{g}_z \notin \langle \tilde{g}_1, \ldots, \tilde{g}_k \rangle$. For a contradiction, let us assume $\tilde{g}_z \in \langle \tilde{g}_1, \ldots, \tilde{g}_k \rangle$. Thus, there exists $g \in \langle g_1, \ldots, g_k \rangle$ such that $\langle a| g_z |a\rangle = \langle a| g |a\rangle$. This implies that $\langle a| g_z g |a\rangle = I^{\otimes n-1}$ and hence $g_z g = (-1)^a Z_j$. But $(-1)^a Z_j$ does not commute with g_x and $\langle G \rangle$ is a commuting group containing $g_z g$ and g_x resulting in a contradiction. □

We are ready to prove prove Lemma 4.

Proof. Let us define the generating set $G^{(0)} = \{g_1, \ldots, g_{n+t}\} \in \mathcal{G}(n+t, n+t)$ by $g_j := V Z_j V^\dagger$ where Z_j is $I^{\otimes n+t}$ with the j^{th} tensor factor replaced by Z. Then, from Eq. (30b), it is evident that $\Pi_{G^{(0)}} = V |0\rangle\langle 0|^{\otimes n+t}_{\text{abc}} V^\dagger$. Substituting into the LHS of Eq. (31a), we get:

$$\text{Tr}_{\text{ab}} \left(V |0\rangle\langle 0|^{\otimes n+t}_{\text{abc}} V^\dagger |x\rangle\langle x|_{\text{a}} \right) = 2^{-(n+t)} \sum_{g \in \langle G^{(0)} \rangle} \omega(g) \text{Tr}\left(|g|_{\text{a}} |x\rangle\langle x|_{\text{a}}\right) \text{Tr}\left(|g|_{\text{b}}\right) |g|_{\text{c}} \,. \tag{A1}$$

From the RHS of Eq. (A1), we note that terms associated with g will be zero if certain constraints on g are not satisfied. In particular, for a fixed $g \in \langle G^{(0)} \rangle$ to produce a non-zero contribution to the sum, it is necessary that:

- Register 'a' constraints: for all $j \in [w]$, $|g|_j \in \{I, Z\}$
- Register 'b' constraints: for all $j \in [n-w]$, $|g|_{w+j} = I$.

We note that for any $G \in \mathcal{G}(n,k)$ and $j \in [n]$ the sets $S_{\mathrm{a}}(G,j) := \left\{ g \in \langle G \rangle \,|\, |g|_j \in \{I,Z\} \right\}$ and $S_{\mathrm{b}}(G,j) := \left\{ g \in \langle G \rangle \,|\, |g|_j = I \right\}$ are both subgroups of $\langle G \rangle$. Hence these are generated by some generating sets $G_{\mathrm{a}}(G,j)$ and $G_{\mathrm{b}}(G,j)$ respectively. Through a procedure similar to performing computational basis measurements on a stabilizer state in the Gottesman-Knill theorem, the ConstrainStabs algorithm computes the generating set $\tilde{G}$.

Starting from the stabilizer group $\langle G^{(0)} \rangle$, the ConstrainStabs algorithm imposes the above constraints to find a stabilizer generating set $\tilde{G} = \left\{ \tilde{g}_1, \ldots, \tilde{g}_{\tilde{k}} \right\}$ that satisfies the properties:

1. $\langle \tilde{G} \rangle \subseteq \langle G^{(0)} \rangle$
2. for all $g \in \langle G^{(0)} \rangle$, g satisfies the register 'a' and 'b' constraints if and only if $g \in \langle \tilde{G} \rangle$.

This allows us to replace the sum over $g \in \langle G^{(0)} \rangle$ by a sum over $g \in \langle \tilde{G} \rangle$ in the RHS of Eq. (A1). Some further manipulation gives:

$$\begin{aligned} 2^{-(n+t)} \sum_{g \in \langle \tilde{G} \rangle} \omega(g) \mathrm{Tr}\left(|g|_a |x\rangle\langle x|_a \right) \mathrm{Tr}\left(|g|_b \right) |g|_c &= 2^{-(n+t)} \left| \langle \tilde{G} \rangle \right| \mathrm{Tr}_{\mathrm{ab}}\left(\Pi_{\tilde{G}} |x\rangle\langle x|_{\mathrm{a}} \right) \\ &= 2^{-n-t+\tilde{k}} \mathrm{Tr}_{\mathrm{ab}}\left(\Pi_{\tilde{G}} |x\rangle\langle x|_{\mathrm{a}} \right), \end{aligned} \tag{A2}$$

where $\tilde{k} := \left| \tilde{G} \right|$. We will later define r and v such that $\tilde{k} = t - r + v$ thus proving Eq. (31a).

We now define the linear map $f_x : \langle \tilde{G} \rangle \to \mathcal{P}_t$ by:

$$f_x(g) = 2^{-(n-w)} \mathrm{Tr}_{\mathrm{ab}}\left(g |x\rangle\langle x|_{\mathrm{a}} \right) \tag{A3}$$

$$= \omega(g) \langle x | |g|_{\mathrm{a}} |x\rangle |g|_{\mathrm{c}}. \tag{A4}$$

We show that this is a group homomorphism. Let $g, g' \in \langle \tilde{G} \rangle$, then:

$$\begin{aligned} f_x(g) f_x(g') &= \omega(g)\omega(g') \langle x | |g|_{\mathrm{a}} |x\rangle\langle x| |g'|_{\mathrm{a}} |x\rangle |g|_{\mathrm{c}} |g'|_{\mathrm{c}} \\ &= \omega(g)\omega(g') \langle x | |g|_{\mathrm{a}} |g'|_{\mathrm{a}} |x\rangle |g|_{\mathrm{c}} |g'|_{\mathrm{c}} \\ &= 2^{-(n-w)} \mathrm{Tr}_{\mathrm{ab}}\left(gg' |x\rangle\langle x|_{\mathrm{a}} \right) \\ &= f_x(gg') \end{aligned}$$

where in the second equality, we used the fact that $|g'|_a$ commutes with $|x\rangle\langle x|$ since g' satisfies the register 'a' constraints.

This shows that the image $f_x(\langle \tilde{G} \rangle)$ is an Abelian subgroup of $\mathcal{P}_t$ generated by the set $\left\{ f_x(g) \mid g \in \tilde{G} \right\}$. Starting from Eq. (A2), we now note that:

$$2^{-n-t+\tilde{k}} \mathrm{Tr}_{\mathrm{ab}}\left(\Pi_{\tilde{G}} |x\rangle\langle x|_{\mathrm{a}} \right) = 2^{-t+\tilde{k}-w} f_x(\Pi_{\tilde{G}}) \tag{A5}$$

where we have extended the domain of f_x by linearity i.e.

$$f_x(\Pi_{\tilde{G}}) = \left| \langle \tilde{G} \rangle \right|^{-1} \sum_{g \in \tilde{G}} f_x(g). \tag{A6}$$

We note that the list of elements $f_x(\tilde{g}_1), \ldots, f_x(\tilde{g}_{\tilde{k}})$ that generate $f_x(\langle \tilde{G} \rangle)$ may be dependent. This can happen if and only if there is a $g \in \langle \tilde{G} \rangle \setminus \{ I^{\otimes n+t} \}$ such that $f_x(g) = I^{\otimes t}$. Further, we note that the group $f_x(\langle \tilde{G} \rangle)$ may contain the element $-I^{\otimes t}$. If this is the case, it is easy to show that $f_x(\Pi_{\tilde{G}}) = 0$.

We now outline a simple procedure that allows us to:

1. Identify constraints on x that are necessary to ensure that $-I^{\otimes t} \notin f_x(\langle \tilde{G} \rangle)$ and, assuming x satisfies all such constraints,
2. Identify a subset of $\langle \tilde{G} \rangle$ such that its image under f_x is independent and generates $f_x(\langle \tilde{G} \rangle)$.

Let us start with the set $\tilde{G}^{(0)} := \tilde{G}$ and for each $j \in [w]$ we update $\tilde{G}^{(0)} \to \tilde{G}^{(1)} \to \ldots \to \tilde{G}^{(w)}$. On, the j^{th} update procedure we:

1. Put $\tilde{G}^{(j-1)}$ into $ZX(j)$-form.
2. If $\tilde{G}^{(j-1)}$ does not have a leading generator, $\tilde{G}^{(j)} \leftarrow \tilde{G}^{(j-1)}$
3. If $\tilde{G}^{(j-1)}$ has a leading generator, then it must be a leading-Z. Call this g_z.
4. Map all elements $g \in \tilde{G}^{(j-1)} \setminus \{g_z\}$ to $\langle x_1 \ldots x_j | \, g \, | x_1 \ldots x_j \rangle$ where the $|x_1 \ldots x_j\rangle$ vectors act on the first j qubits. Check if the group generated by these elements, $\langle G(x,j) \rangle$, contains either $\langle 0| \langle x_1 \ldots x_{j-1}| \, g_z \, |x_1 \ldots x_{j-1}\rangle |0\rangle$ or $-\langle 0| \langle x_1 \ldots x_{j-1}| \, g_z \, |x_1 \ldots x_{j-1}\rangle |0\rangle$ (where the $|0\rangle$ vectors act on qubit j and the $|x_1 \ldots x_{j-1}\rangle$ vectors act on the first $j-1$ qubits). If neither is true then $\tilde{G}^{(j)} \leftarrow \tilde{G}^{(j-1)}$.
5. If the group $\langle G(x,j) \rangle$ contains $(-1)^a \langle 0| \langle x_1 \ldots x_{j-1}| \, g_z \, |x_1 \ldots x_{j-1}\rangle |0\rangle$ for either $a = 0$ or $a = 1$, then we require $x_j = a$. In either case ($x_j = 0$ or $x_j = 1$), we note that $f_x(g_z)$ is dependent on $\left\{ f_x(g) \mid g \in \tilde{G}^{(j-1)} \setminus \{g_z\} \right\}$ hence we set $\tilde{G}^{(j)} \leftarrow \tilde{G}^{(j-1)} \setminus \{g_z\}$.

It is clear that for all j where Step 5 applies, our constraint on x_j is necessary to ensure that $-I^{\otimes t} \notin f_x(\langle \tilde{G} \rangle)$.

The final output $\tilde{G}^{(w)}$ of this procedure is a subset of $\langle \tilde{G} \rangle$ such that its image under f_x is independent and generates $f_x(\langle \tilde{G} \rangle)$. To see that $\left\{ f_x(g) \mid g \in \tilde{G}^{(w)} \right\}$ generates $f_x(\langle \tilde{G} \rangle)$, we note that the above procedure only deletes elements in Step 5. In this case, the deleted element g_z has the property that $f_x(g_z)$ is dependent on the f_x image of the remaining elements. Hence $\left\{ f_x(g) \mid g \in \tilde{G}^{(w)} \right\}$ generates $f_x(\langle \tilde{G} \rangle)$ since $\left\{ f_x(g) \mid g \in \tilde{G}^{(0)} \right\}$ generates $f_x(\langle \tilde{G} \rangle)$. We can see that $\left\{ f_x(g) \mid g \in \tilde{G}^{(w)} \right\}$ is independent by induction. First, we note that $\left\{ g \in \tilde{G}^{(0)} \right\}$ is independent. Now for the j^{th} induction step, assume that $\left\{ \langle x_1 \ldots x_{j-1}| \, g \, |x_1 \ldots x_{j-1}\rangle \mid g \in \tilde{G}^{(j-1)} \right\}$ is independent (where the vector $|x_1 \ldots x_{j-1}\rangle$ acts on the first $j-1$ qubits). Then to see that $\left\{ \langle x_1 \ldots x_j| \, g \, |x_1 \ldots x_j\rangle \mid g \in \tilde{G}^{(j)} \right\}$ is independent let us, without loss of generality, assume $\tilde{G}^{(j-1)}$ is in ZX-Form with respect to qubit j. Then it is clear that $\left\{ \langle x_1 \ldots x_j| \, g \, |x_1 \ldots x_j\rangle \mid g \in \tilde{G}^{(j-1)}, g \text{ not a leading generator} \right\}$ are independent. Thus dependence can only arise if $\langle x_1 \ldots x_j| \, g_z \, |x_1 \ldots x_j\rangle$ is dependent on $\left\{ \langle x_1 \ldots x_j| \, g \, |x_1 \ldots x_j\rangle \mid g \in \tilde{G}^{(j-1)}, g \text{ not a leading generator} \right\}$. We explicitly check this (in Step 4) and exclude g_z from $\tilde{G}^{(j)}$ if it gives rise to dependence. Hence by induction we have shown that $\left\{ \langle x| \, g \, |x\rangle \mid g \in \tilde{G}^{(w)} \right\}$ is independent. This immediately leads to the independence of $\left\{ f_x(g) \mid g \in \tilde{G}^{(w)} \right\}$.

We define $G := f_x(\tilde{G}^{(w)}) \in \mathcal{G}(t,k)$ for some $k \leq t$. Using Eq. (A5) and the definition of $\tilde{k}$, Eq. (31b) follows. We define $r := t - k$ and define v as the number of rows deleted in Step 5. Hence, $\left|\tilde{G}\right| = \left|\tilde{G}^{(w)}\right| + v = |G| + v = t - r + v$. Thus, it is clear that $v \in \{0, 1, \ldots, w\}$ and $r \leq t$. We now show that $r \leq n - w$. Let us note that to derive $\tilde{G}$ from $G^{(0)}$, we imposed the constraints on registers 'a' and 'b'. Since $G^{(0)}$ is a maximal generating set, just imposing the register 'b' constraints must result in the deletion of between $(n-w)$ and $2(n-w)$ generators. Subsequently imposing register 'a' constraints must result in the further deletion of d_{a} generators where, by using Lemma 11, one can show that d_{a} is between 0 and $w - v$. Thus,

$$\begin{aligned}
\left|\tilde{G}\right| &\in \left\{ \left|G^{(0)}\right| - 2(n-w) - (w-v), \left|G^{(0)}\right| - 2(n-w) - w + 1 \ldots, \left|G^{(0)}\right| - (n-w) \right\} \\
&= \{(n+t) - 2(n-w) - (w-v), (n+t) - 2(n-w) - (w-v) + 1 \ldots, (n+t) - (n-w)\} \\
&= \{-n + t + w + v, -n + t + w + v + 1 \ldots, t + w\}
\end{aligned} \tag{A7}$$

Now, $\left|\tilde{G}\right| = t - r + v$ by definition of r and v. So:

$$\begin{aligned}
r &\in \{(t+v) - (t+w), \ldots, (t+v) - (-n + t + w + v)\} \\
&= \{v - w, \ldots, n - w\}.
\end{aligned} \tag{A8}$$

This shows that $r \leq n - w$ completing our proof.

□

Appendix B: Proof of Lemma 5

Proof. We will provide a sequence of gates that forms a unitary W transforming Π_G into $|0\rangle\langle 0|^{\otimes t-r} \otimes I^{\otimes r}$ for an arbitrary stabilizer generator matrix $G \in \mathcal{G}(t, t-r)$. Equivalently, it means that W should transform a generator matrix G into a generator matrix corresponding to a projector $|0\rangle\langle 0|^{\otimes t-r} \otimes I^{\otimes r}$, i.e.,

$$G_0 = [\ X \parallel Z\ ] \quad \xrightarrow{W} \quad G_{fin} = [\ 0 \parallel I \mid 0\ ]\,. \tag{B1}$$

Here, we will adopt the formulation in terms of tableaux given in Ref. [16]. For our algorithm we only require the stabilizers, not the destabilisers so the matrices given in equation B1 (and in the sequel) correspond to the lower half of the destabiliser+stabiliser tableaux of Aaronson and Gottesman.

Each of $(t-r)$ rows of a generator matrix corresponds to a stabilizer generator given by a Pauli matrix encoded as a binary vector of length $2t$. The first t entries correspond to X stabilizers (i.e. if the entry in column $j \in [t]$ equals one, then there is a Pauli X acting on the j^{th} qubit), while the remaining t entries correspond to Z stabilizers. If a row k has a qubit with a 1 in both the X and Z portion then the stabiliser has the operator Y_k on qubit k (this differs from $X_k Z_k$ by a factor of i). We use the symbol $\parallel$ to visually separate the X and Z parts, and the separator $\mid$ to separate a square block within each part. We also use X and Z to denote arbitrary entries in the corresponding parts, and I to denote a square $(t-r)\times(t-r)$ identity matrix. In addition to the Pauli matrices each stabiliser has a phase ± 1 associated to it. This information is stored in a binary vector f of length $(t-r)$.

First, given G_0, we can perform row sums, as they correspond to stabiliser multiplication. We can also swap pairs of columns j and $j+t$ (one in X part, the other in Z part) by applying a Hadamard gate to qubits j. Using these two operations, we can bring the X part to the reduced row echelon form. More precisely, using row sums we can perform Gaussian elimination of the X part, and each time we find a column with no leading 1 in it, we can bring the missing 1 from the Z part (if one exists) using a Hadamard gate. Since the rows are independent we will obtain exactly $(n-r)$ leading 1s in this way. Finally, using $SWAP$ gates (each composed of three CX gates), we can permute the columns so that the first $(t-r)\times(t-r)$ block in the X part is given by the identity matrix. Thus, after using at most $t-r$ Hadamard gates and $3(t-r)$ instances of CX gates, we arrive at

$$G_1 = [\ I \mid X \parallel Z\ ]. \tag{B2}$$

Next, we can use CX gates to clear the remaining $(t-r)\times r$ block of the X part. Specifically, if there is a 1 in row j and column k of this block, the application of CX controlled on qubit j and targeted on qubit k flips that 1 to 0. It also non-trivially affects the entries of the Z part, but we will deal with that in the next step. Thus, after using at most $(t-r)r$ instances of CX gates, we obtain

$$G_2 = [\ I \mid 0 \parallel Z\ ]. \tag{B3}$$

The third step employs phase gates to ensure that the main diagonal of the Z part, i.e. the elements Z_{jj}, are all zero. To achieve this, it is enough to apply S to every qubit j such that $Z_{jj} = 1$. This requires the use of at most $(t-r)$ phase gates and results in

$$G_3 = [\ I \mid 0 \parallel \tilde{Z}\ ], \tag{B4}$$

with tilde indicating the zero diagonal of the main $(t-r)\times(t-r)$ block in the Z part.

Now, we can use CZ gates to clear the last $(t-r)\times r$ block of the Z part. Specifically, if there is a 1 in row j and column k of this block, the application of CZ controlled on qubit j and targeted on qubit k flips that 1 to 0. It also does not affect the entries of the X part at all. Thus, after using at most $(t-r)r$ instances of CZ gates, we obtain

$$G_4 = [\ I \mid 0 \parallel \tilde{Z} \mid 0\ ]. \tag{B5}$$

In the fifth step, we employ the fact that stabilisers commute. Assume that there is a non-zero element $\tilde{Z}_{ij} = 1$, $i \neq j$ in the $\tilde{Z}$ matrix, this means there is Pauli Z matrix on qubit j in stabiliser i. Stabiliser i has to commute with stabiliser j, since there is a 1 in element (j,j) of the X block there is a Pauli X acting on qubit j in stabiliser j which would lead to stabiliser i anti-commuting with stabiliser j. Since the X part of the tableau is $[\ I \mid 0\ ]$ we must also have a Pauli Z in stabiliser j acting on qubit i, $\tilde{Z}_{ij} = 1 \implies \tilde{Z}_{ji} = 1$. Repeating this argument with the initial assumption $\tilde{Z}_{ij} = 0$ proves that $\tilde{Z}$ is symmetric $\tilde{Z}^T = \tilde{Z}$.

This allows us to zero the whole Z part using CZ gates. More precisely, for each unordered pair (j,k) such that $\tilde{Z}_{jk} = \tilde{Z}_{kj} = 1$, the application of a CZ gate controlled on qubit j and targeted on qubit k flips both those 1's to 0's. It also does not affect any other entries. Thus, after using at most $r(r-1)/2$ instances of CZ gates, we arrive at

$$G_5 = [\ I \mid 0 \parallel 0\ ]. \tag{B6}$$

At this stage it is convenient to zero the phase vector f. For each row k if $p_k = 1$ we apply $Z_k = S_k^2$. Due to the form of the X part of the matrix it is easy to see that this will multiply stabiliser k by -1 and leave all the others invariant. This requires at most $2(t-r)$ applications of the S gate.

The final step requires implementing $t-r$ Hadamard gates to transform the above G_5 to G_{fin}. Summarising, in all steps we used at most $2(4+r)t - r(17+3r)/2$ Clifford gates including at most $2(t-r)$ Hadamard gates.

□

Appendix C: Proof of Lemma 6

Proof. From the definition of $|\psi(y)\rangle$, Eq. (48), we have

$$\||\psi(y)\rangle\|_2^2 := \xi^* \cdot 2^{t-r+v-w} \left\| \langle 0|^{\otimes t-r} W |\tilde{y}\rangle \right\|_2^2, \tag{C1}$$

Now, combining the statements of Lemma 4 and Lemma 5 we have:

$$\mathrm{Tr}_{\mathrm{ab}} \left(V |0\rangle\langle 0|_{\mathrm{abc}}^{\otimes n+t} V^\dagger |x\rangle\langle x|_{\mathrm{a}} \right) = 2^{-r+v-w} W^\dagger (|0\rangle\langle 0|^{\otimes t-r} \otimes I^{\otimes r}) W. \tag{C2}$$

and so

$$2^t \left\| \langle x|_{\mathrm{a}} \langle \tilde{y}|_{\mathrm{c}} V |0\rangle_{\mathrm{abc}}^{\otimes n+t} \right\|_2^2 = 2^{t-r+v-w} \left\| \langle 0|^{\otimes t-r} W |\tilde{y}\rangle \right\|_2^2. \tag{C3}$$

Therefore,

$$\||\psi(y)\rangle\|_2^2 = \xi^* \cdot 2^t \left\| \langle x|_{\mathrm{a}} \langle \tilde{y}|_{\mathrm{c}} V |0\rangle_{\mathrm{abc}}^{\otimes n+t} \right\|_2^2 \leq \xi^* \cdot 2^t \left\| \langle \tilde{y}|_{\mathrm{c}} V |0\rangle_{\mathrm{abc}}^{\otimes n+t} \right\|_2^2. \tag{C4}$$

Let us take a closer look at $\langle \tilde{y}|_{\mathrm{c}} V |0\rangle_{\mathrm{abc}}^{\otimes n+t}$. Referring to Fig. 5, recall that the unitary V describes a Clifford circuit of the form

$$V = C_t \prod_{j=1}^{t} CX_j C_{j-1} \tag{C5}$$

where C_j is an arbitrary Clifford gate on n qubits in register 'ab', CX_j is a CNOT gate controlled on one of the qubits from register 'ab' and targeted at the j-th qubit in register 'c', and the product is ordered from right to left (i.e., the rightmost term is given by C_0). We then have

$$\langle \tilde{y}|_{\mathrm{c}} V |0\rangle_{\mathrm{abc}}^{\otimes n+t} = \langle \tilde{y}_1 \dots \tilde{y}_t|_{\mathrm{c}} \left(C_t \prod_{j=1}^{t} CX_j C_{j-1} \right) |0\rangle_{\mathrm{ab}}^{\otimes n} \otimes |0\rangle_{\mathrm{c}}^{\otimes t} \tag{C6}$$

$$= \langle \tilde{y}_2 \dots \tilde{y}_t|_{\mathrm{c}} \left(C_t \prod_{j=2}^{t} CX_j C_{j-1} \right) |\Phi_1\rangle_{\mathrm{ab}} \otimes |0\rangle_{\mathrm{c}}^{\otimes (t-1)} \tag{C7}$$

with

$$|\Phi_1\rangle_{\mathrm{ab}} = \langle \tilde{y}_1|_{\mathrm{c}} CX_1 C_0 |0\rangle_{\mathrm{ab}}^{\otimes n} \otimes |0\rangle_{\mathrm{c}} \tag{C8}$$

being an n-qubit unnormalised stabliser state (note that we could commute the projector $\langle \tilde{y}_1|_{\mathrm{c}}$ through the circuit as the first qubit in the register 'c' is never again affected by it). In order to normalize $|\Phi_1\rangle_{\mathrm{ab}}$ we first write

$$C_0 |0\rangle_{\mathrm{ab}}^{\otimes n} = c_0 |0\psi_0\rangle_{\mathrm{ab}} + |1\psi_1\rangle_{\mathrm{ab}}, \tag{C9}$$

with $|c_0|^2 + |c_1|^2 = 1$ and the distinguished first qubit corresponding to the control qubit of CX_1. We then use the above to obtain

$$|\Phi_1\rangle_{\mathrm{ab}} = \langle \tilde{y}_1|_{\mathrm{c}} CX_1 (c_0 |0\psi_0\rangle_{\mathrm{ab}} + c_1 |1\psi_1\rangle_{\mathrm{ab}}) \otimes |0\rangle_{\mathrm{c}} \tag{C10}$$

$$= \langle \tilde{y}_1|_{\mathrm{c}} (c_0 |0\psi_0\rangle_{\mathrm{ab}} \otimes |0\rangle_{\mathrm{c}} + c_1 |1\psi_1\rangle_{\mathrm{ab}} \otimes |1\rangle_{\mathrm{c}}) \tag{C11}$$

$$= \frac{1}{\sqrt{2}} (c_0 |0\psi_0\rangle_{\mathrm{ab}} + c_1 (-i)^{\tilde{y}_1} |1\psi_1\rangle_{\mathrm{ab}}), \tag{C12}$$

and so we conclude that the normalised state is given by

$$|\Phi_1'\rangle_{\mathrm{ab}} = \frac{1}{\sqrt{2}} |\Phi_1\rangle_{\mathrm{ab}} \,. \tag{C13}$$

We thus have

$$\langle \tilde{y}|_{\mathrm{c}} \, V \, |0\rangle_{\mathrm{abc}}^{\otimes n+t} = \frac{1}{\sqrt{2}} \langle \tilde{y}_2 \dots \tilde{y}_t|_{\mathrm{c}} \left(C_t \prod_{j=2}^{t} C X_j C_{j-1} \right) |\Phi_1'\rangle_{\mathrm{ab}} \otimes |0\rangle_{\mathrm{c}}^{\otimes (t-1)} \,, \tag{C14}$$

and we can repeat the whole procedure again. More precisely, we introduce an n-qubit unnormalised stabiliser state

$$|\Phi_2\rangle_{\mathrm{ab}} = \langle \tilde{y}_2|_{\mathrm{c}} \, C X_2 C_1 \, |\Phi_1'\rangle_{\mathrm{ab}} \otimes |0\rangle_{\mathrm{c}} \,, \tag{C15}$$

we decompose $C_1 \, |\Phi_1'\rangle_{\mathrm{ab}}$ analogously as we did in Eq. (C9), and repeating the same reasoning we arrive at

$$\langle \tilde{y}|_{\mathrm{c}} \, V \, |0\rangle_{\mathrm{abc}}^{\otimes n+t} = \frac{1}{\sqrt{2^2}} \langle \tilde{y}_3 \dots \tilde{y}_t|_{\mathrm{c}} \left(C_t \prod_{j=3}^{t} C X_j C_{j-1} \right) |\Phi_2'\rangle_{\mathrm{ab}} \otimes |0\rangle_{\mathrm{c}}^{\otimes (t-2)} \,. \tag{C16}$$

Repeating it t times in total we finally arrive at

$$\langle \tilde{y}|_{\mathrm{c}} \, V \, |0\rangle_{\mathrm{abc}}^{\otimes n+t} = \frac{1}{\sqrt{2^t}} C_t \, |\Phi_t'\rangle_{\mathrm{ab}} \,. \tag{C17}$$

Substituting the above to the inequality from Eq. (C4) we finally arrive at

$$\| |\psi(y)\rangle \|_2^2 \leq \xi^* . \tag{C18}$$

□

Appendix D: Proof of Lemma 7

In order to prove our result, we will need the definition of a *very-weak martingale* and a theorem from Ref. [40] given below.

Definition 12 (Very-weak martingale)**.** Let $N \in \mathbb{N}$, Ω be a sample space and for all $j \in \mathbb{N}$, let $X_j : \Omega \to \mathbb{R}^N$ be a random variable taking values in $\mathbb{R}^N$ such that $X_0 = 0$, $\mathbb{E}\left[\|X_j\|_2\right] < \infty$ and $\mathbb{E}\left[X_j \mid X_{j-1}\right] = X_{j-1}$. Then we call the sequence $(X_0, X_1, \dots)$ a *very-weak martingale in* $\mathbb{R}^N$.

Theorem 13 (Theorem 1.8 of Ref. [40])**.** *Let X ba a very-weak martingale taking values in $\mathbb{R}^N$ such that $X_0 = 0$ and for every j, $\|X_j - X_{j-1}\|_2 \leq 1$. Then for every $a > 0$:*

$$\Pr\left(\|X_s\|_2 \geq a\right) \leq 2e^{1-(a-1)^2/2s} < 2e^2 \exp\left(-a^2/2s\right) . \tag{D1}$$

We can now prove Lemma 7.

Proof. The proof is a simple application of Theorem 13. Let us use $R : \mathbb{C}^d \to \mathbb{R}^{2d}$ to denote the two-norm preserving linear map $R(a_1 + ib_1, \dots, a_d + ib_d) = (a_1, b_1, \dots, a_d, b_d)$. For $s \in \mathbb{N}$, we define the random variable $Y_s \in \mathbb{R}^{2d}$ as follows: $Y_0 = (0, \dots, 0)$ and for $s > 0$:

$$Y_s := \frac{s}{\sqrt{m} + \sqrt{p}} R\left(|\overline{\psi}\rangle - |\mu\rangle\right) , \tag{D2}$$

where we note that $|\overline{\psi}\rangle$ depends on s as per Eq. (52).

We now note that Y_s is a very-weak martingale since $Y_0 = 0$ and

$$\mathbb{E}\left[\|Y_s\|_2\right] = \frac{s}{\sqrt{m} + \sqrt{p}} \mathbb{E}\left[\sqrt{\left(\langle\overline{\psi}| - \langle\mu|\right)\left(|\overline{\psi}\rangle - |\mu\rangle\right)}\right] < \infty, \tag{D3}$$

as well as

$$\mathbb{E}\left[Y_s \mid Y_{s-1}\right] = \mathbb{E}\left[Y_{s-1} + \frac{1}{\sqrt{m}+\sqrt{p}} R\left(|\psi_{x_s}\rangle - |\mu\rangle\right) \mid Y_{s-1}\right] = Y_{s-1}. \tag{D4}$$

Additionally, we note that $\|Y_s - Y_{s-1}\|_2 \leq 1$, since:

$$\|Y_s - Y_{s-1}\|_2 = \left\| \frac{1}{\sqrt{m}+\sqrt{p}} R\left(|\psi_{x_s}\rangle - |\mu\rangle\right) \right\|_2 \tag{D5}$$

$$= \frac{1}{\sqrt{m}+\sqrt{p}} \sqrt{\langle\psi_{x_s}|\psi_{x_s}\rangle - \langle\psi_{x_s}|\mu\rangle - \langle\mu|\psi_{x_s}\rangle + \langle\mu|\mu\rangle} \tag{D6}$$

$$\leq \frac{1}{\sqrt{m}+\sqrt{p}} \sqrt{m + 2\sqrt{mp} + p} = 1. \tag{D7}$$

Hence, by Theorem 13:

$$\Pr\left(\|Y_s\|_2 \geq a\right) = \Pr\left(\left\| |\overline{\psi}\rangle - |\mu\rangle \right\|_2 \geq \frac{a(\sqrt{m}+\sqrt{p})}{s} \right) < 2e^2 \exp\left(-a^2/2s\right). \tag{D8}$$

Substituting $\epsilon = a(\sqrt{m}+\sqrt{p})/s$ proves Eq. (53).

To prove Eq. (54), we define:

$$|\Delta\rangle := |\mu\rangle - |\overline{\psi}\rangle, \tag{D9}$$

and note that

$$\left\| |\overline{\psi}\rangle\langle\overline{\psi}| - |\mu\rangle\langle\mu| \right\|_1 = \left\| |\mu\rangle\langle\Delta| + |\Delta\rangle\langle\mu| - |\Delta\rangle\langle\Delta| \right\|_1 \leq \left\| |\mu\rangle\langle\Delta| \right\|_1 + \left\| |\Delta\rangle\langle\mu| \right\|_1 + \left\| |\Delta\rangle\langle\Delta| \right\|_1 \tag{D10}$$

$$= 2\left\| |\mu\rangle \right\|_2 \left\| |\Delta\rangle \right\|_2 + \left\| |\Delta\rangle \right\|_2^2 = \left\| |\Delta\rangle \right\|_2 \left(\left\| |\Delta\rangle \right\|_2 + 2\sqrt{p} \right). \tag{D11}$$

Now, employing the above, if $\left\| |\Delta\rangle \right\|_2 \leq \epsilon$ then $\left\| |\overline{\psi}\rangle\langle\overline{\psi}| - |\mu\rangle\langle\mu| \right\|_1 \leq \epsilon(\epsilon + 2\sqrt{p})$. Applying this observation to the already proven Eq. (53) yields:

$$\Pr\left(\left\| |\overline{\psi}\rangle\langle\overline{\psi}| - |\mu\rangle\langle\mu| \right\|_1 \geq \epsilon(\epsilon + 2\sqrt{p}) \right) \leq 2e^2 \exp\left(\frac{-s\varepsilon^2}{2(\sqrt{m}+\sqrt{p})^2} \right).$$

We can now define a new variable $\varepsilon = \epsilon(\epsilon + 2\sqrt{p})$ and solve this quadratic equation for ϵ. Taking only the positive solution gives $\epsilon = \sqrt{p+\varepsilon} - \sqrt{p}$, which immediately leads to Eq. (54).

□

Appendix E: CH form

Following the formalism developed in Ref. [30], any stabilizer state $|\sigma\rangle$ of n qubits can be written as

$$|\sigma\rangle = \omega U_C U_H |s\rangle, \tag{E1}$$

where U_C is the control type operator (in our case effectively meaning that it consists of products of S, CX and CZ gates), U_H is the Hadamard-type operator (consisting only of products of H gates), s is a bit string of length n representing one of the computational basis states, and ω is a complex number. The unitary U_C is fully specified by three $n \times n$ matrices F, G, M with entries in $\mathbb{Z}_2$ and a phase vector γ of length n with entries in $\mathbb{Z}_4$. Together, they describe the action of U_C on Pauli matrices:

$$U_C^\dagger Z_j U_C = \prod_{k=1}^{n} Z_k^{G_{jk}}, \quad U_C^\dagger X_j U_C = i^{\gamma_j} \prod_{k=1}^{n} X_k^{F_{jk}} Z_k^{M_{jk}}, \tag{E2}$$

where X_j and Z_j are Pauli matrices acting on the j-th qubit. The unitary U_H is fully specified by a bit string v of length n, with $v_j = 1$ if U_H acts with a Hadamard on the j-th qubit. Thus, a general stabilizer state $|\sigma\rangle$ of n qubits is described by a tuple $\{F, G, M, \gamma, v, s, \omega\}$.

The initial state is represented by the following tuple

$$|0\rangle^{\otimes n} \quad \Longleftrightarrow \quad \{F=\mathbb{1}, G=\mathbb{1}, M=0, \gamma=0, v=0, s=0, \omega=1\}. \tag{E3}$$

The authors of Ref. [30] found an efficient way to find the tuple $\{F', G', M', \gamma', v', s', \omega'\}$ representing $V|0\rangle^{\otimes n}$ for an arbitrary Clifford unitary V. The run-time of this evolution subroutine scales polynomially with the total number of qubits n: the "C-type" gates (S, CX, CZ) have linear time complexity $O(n)$, while applying a Hadamard gate takes $O(n^2)$ steps. For completeness, we include the update rules for left, $\mathcal{L}[\Gamma]$, and right, $\mathcal{R}[\Gamma]$, multiplication of U_C by a C-type unitary Γ. All phase vector updates are performed modulo four, and each update containing the symbol p should be read as applying to all $p \in \{1, \ldots, n\}$ in turn.

$$\mathcal{R}[\mathrm{S_q}] : \begin{cases} M_{p,q} & \leftarrow & M_{p,q} \oplus F_{p,q} \\ \gamma_p & \leftarrow & \gamma_p - F_{p,q} \end{cases} \qquad \mathcal{L}[\mathrm{S_q}] : \begin{cases} M_{q,p} & \leftarrow & M_{q,p} \oplus G_{q,p} \\ \gamma_q & \leftarrow & \gamma_q - 1 \end{cases} \tag{E4a}$$

$$\mathcal{R}[\mathrm{CZ_{q,r}}] : \begin{cases} M_{p,q} & \leftarrow & M_{p,q} \oplus F_{p,r} \\ M_{p,r} & \leftarrow & M_{p,r} \oplus F_{p,q} \\ \gamma_p & \leftarrow & \gamma_p + 2F_{p,q}F_{p,r} \end{cases} \qquad \mathcal{L}[\mathrm{CZ_{q,r}}] : \begin{cases} M_{q,p} & \leftarrow & M_{q,p} \oplus G_{r,p} \\ M_{r,p} & \leftarrow & M_{r,p} \oplus G_{q,p} \end{cases} \tag{E4b}$$

$$\mathcal{R}[\mathrm{CX_{q,r}}] : \begin{cases} G_{p,q} & \leftarrow & G_{p,q} \oplus G_{p,r} \\ F_{p,r} & \leftarrow & F_{p,r} \oplus F_{p,q} \\ M_{p,q} & \leftarrow & M_{p,q} \oplus M_{p,r} \end{cases} \qquad \mathcal{L}[\mathrm{CX_{q,r}}] : \begin{cases} \gamma_q & \leftarrow & \gamma_q + \gamma_r + 2(MF^T)_{q,r} \\ G_{r,p} & \leftarrow & G_{r,p} \oplus G_{q,p} \\ F_{q,p} & \leftarrow & F_{q,p} \oplus F_{r,p} \\ M_{q,p} & \leftarrow & M_{q,p} \oplus M_{r,p} \end{cases} \tag{E4c}$$

We note a slight difference to the update rules as presented by the authors of Ref. [30]: in the update rule for $\mathcal{L}[\mathrm{CX_{q,r}}]$ we update γ before updating F and M to emphasise that γ must be updated based on the old values of F and M, rather than the new. It will be significant that the action of the operation $\mathcal{L}[\mathrm{CX_{q,r}}]$ on the F matrix is column addition and that, since this addition is modulo two, the right-action of the swap gate, $\mathrm{CX_{q,r}CX_{r,q}CX_{q,r}}$, on F is just to swap the columns r and q.

We will also employ the equation given in Ref. [30] to compute inner products between CH-form stabiliser states and computational basis states,

$$\langle x| U_C U_H |s\rangle = \langle 0|^{\otimes n} \left(\prod_{p=1}^{n} U_C^{-1} X_p^{x_p} U_C \right) U_H |s\rangle = 2^{-\frac{|v|}{2}} i^{\mu} \prod_{j:\ v_j=1} (-1)^{u_j s_j} \prod_{j:\ v_j=0} \langle u_j | s_j \rangle , \tag{E5}$$

where $u_j = xF$, and $\mu = x \cdot \gamma + 2k$ for a constant $k \in \{0, 1\}$ which may be computed in quadratic time given x and the CH-form data. Indeed, some algebra demonstrates that k may be determined by the relation

$$\prod_{p=1}^{n} U_C^{-1} X_p^{x_p} U_C = i^{x\cdot\gamma} \prod_{p:\ x_p=1} \prod_{j=1}^{n} X_j^{F_{pj}} Z_j^{M_{pj}} = i^{x\cdot\gamma} (-1)^k \prod_{j=1}^{n} \prod_{p:\ x_p=1} Z_j^{M_{pj}} X_j^{F_{pj}}. \tag{E6}$$

Appendix F: Proof of Lemma 9

The proof splits into two parts. First, we show in Lemma 14 that if the CH-form describing the initial $(n+1)$-qubit state $|0\rangle \otimes |\sigma\rangle$ has a certain form, then a CH-form describing $|\sigma\rangle$ may be obtained by simply deleting the first row and column of each F, G and M, and the first element of γ, v and s. We assume the deletion operation takes quadratic time to leading order as an implementation is likely to simply allocate a new $O(n^2)$ sized block of memory and then copy the required values across. Second, we give an algorithm which takes an arbitrary CH-form representing $|0\rangle \otimes |\sigma\rangle$ and outputs in quadratic time a CH-form representing the same state, but in the form required by Lemma 14. We show, in Lemmas 15 and 16, how the CH-form representing such a product state can be brought into a form with at most one qubit k with $v_k = 0$, $s_k = 1$. This is important because we can insert CX gates controlled on any qubit with $v_k = 0$, $s_k = 0$ between U_C and U_H in the CH-form without changing the state (a CX controlled on $|0\rangle$ does nothing). Finally, in Lemma 17, we show that if the CH-form is in the form produced by Lemmas 15 and 16, then inserting CX gates controlled on $|0\rangle$ qubits can bring the CH-form into the form required by Lemma 14.

We label computational basis vectors with bit-strings, denote bitwise addition modulo-2 with the symbol $\oplus$, bitwise multiplication with juxtaposition, and use the operator $:$ to denote concatenation of bitstrings, e.g., if $a = 01$ then $0:a = 001$ and $a:0 = 010$. The first part of the proof is then captured by the following.

Lemma 14. *Consider a stabiliser state $|0\rangle\otimes|\sigma\rangle$ with CH-form $\mathcal{F}=\{F,G,M,\gamma,v,s,\omega\}$ such that: the first column of F has a 1 in the first element and zeros elsewhere, and $s_1=v_1=0$. Then, the CH-form $\mathcal{F}'=\{F',G',M',\gamma',v',s',\omega\}$, where F', G' and M' are formed by deleting the first row and column of F, G and M, respectively, and γ' v' and s' are formed by deleting the first element of γ, v and s, respectively, is a CH-form for $|\sigma\rangle$.*

Proof. Choose an arbitrary n-qubit computational-basis vector $|a\rangle$ and use Eq. (E5) to compute

$$\langle a|\sigma\rangle = \langle 0|0\rangle\langle a|\sigma\rangle = \langle 0{:}a|\,|0\rangle\otimes|\sigma\rangle = \omega\,\langle 0{:}a|\,U_C U_H\,|s\rangle = (-1)^k\omega i^{u\cdot\gamma}\,\langle 0^n|\,X(u)U_H\,|s\rangle \tag{F1}$$

$$= (-1)^k\omega i^{u\cdot\gamma}2^{-|v|/2}\prod_{j:\ v_j=1}(-1)^{u_j s_j}\prod_{j:\ v_j=0}\langle u_j|s_j\rangle\,, \tag{F2}$$

where k is a bit we have introduced to count whether we have swapped an even or odd number of X_j with their Z_j to arrive at the final equality in Eq. (F1). We recall that $u=(0{:}a)F$. If F' is the matrix obtained from F by removing the first row and column, then one can verify that $u=0{:}(aF')$; in particular $u_1=0$. In addition, we define γ', v' and s' by deleting the first element of γ, v and s, respectively, and we let $u'=aF'$. Ignoring k for the moment we go through the rest of the terms in turn. First, since $u_1=0$

$$u'\cdot\gamma' = u\cdot\gamma. \tag{F3}$$

Next, since we have $v_1=0$,

$$|v'| = |v| \tag{F4}$$

and

$$\prod_{j:\ v'_j=1}(-1)^{u'_j s'_j} = \prod_{j:\ v_j=1}(-1)^{u_j s_j}. \tag{F5}$$

Finally, since $\langle u_1|s_1\rangle=1$,

$$\prod_{j:\ v'_j=0}\langle u'_j|s'_j\rangle = \prod_{j:\ j>1,v_j=0}\langle u_j|s_j\rangle = \prod_{j:\ v_j=0}\langle u_j|s_j\rangle\,. \tag{F6}$$

We now turn to the calculation of k. For conciseness we write $x=0{:}a$, and want to simplify

$$X(x)U_C = U_C\prod_{p:\ x_p=1}\left(i^{\gamma_k}\prod_j X_j^{F_{pj}}Z_j^{M_{pj}}\right). \tag{F7}$$

In particular, we will commute all the Z's to the back to write

$$X(x)U_C = (-1)^k i^{u\cdot\gamma}U_C Z(t)X(u). \tag{F8}$$

The bit k may be calculated by the following algorithm. First initialise $k:=0$ and set t to be a length n vector of zeros. Then, for each p with $x_p=1$, we want to compute the product

$$\left(\prod_j X_j^{F_{pj}}Z_j^{M_{pj}})\right)Z(t)X(u). \tag{F9}$$

We update t by adding (mod-2) the p^{th} row of M (i.e., we combine the adjacent Z-type operators), then we commute each X_j through the new $Z(t)$, which gives a (-1) for each j for which both F_{pj} and t_j are non-zero. More explicitly, we update k to be $k+F_p\cdot t(\text{mod }2)$, where F_p is the p^{th} row vector of F.

Since the first column of F has a 1 in the first element and 0 in all others, each F_p for $p>1$ starts with a zero. Therefore, k is not sensitive to the first bit of t except when $p=1$ (since the first element of F_1 is 1). However $x=(0{:}a)$, so $x_1=0$ and so the case $p=1$ does not appear in the product in Eq. (F7). We will therefore compute the same k bit with the "trimmed" data as we would with the original data. □

In order to present the second part of the proof we will need a few lemmas. First, we will prove a useful constraint on the CH-form of a state in the form $|0\rangle\otimes|\sigma\rangle$.

Lemma 15. *Given a stabiliser state,*

$$|0\rangle \otimes |\sigma\rangle = \omega U_C U_H |s\rangle\,, \tag{F10}$$

where the CH-form on the right is defined by the data $\mathcal{F} = \{F, G, M, \gamma, v, s, \omega\}$, at least one of the following is true:

1. *$\omega = 0$;*
2. *$\exists q$ such that $v_q = s_q = 0$;*
3. *$\exists q, r$ $(q \neq r)$ such that $s_q = s_r = 1$ and $v_q = v_r = 0$.*

Proof. We consider the inner product

$$(\langle 1| \otimes \langle a|)\,(|0\rangle \otimes |\sigma\rangle) = \langle 1|0\rangle\,\langle a|\sigma\rangle = 0, \tag{F11}$$

where $a \in \{0,1\}^n$ defines a computational basis vector. We thus have that for all a:

$$0 = \omega\,\langle 1\!:\!a|\,U_C U_H\,|s\rangle\,. \tag{F12}$$

If $\omega = 0$ we are in case 1, otherwise we divide by ω to get

$$0 = \langle 1\!:\!a|\,U_C U_H\,|s\rangle\,. \tag{F13}$$

Applying Eq. (E5) we obtain

$$0 = \langle 0|^{\otimes n}\, X((1\!:\!a)F) U_H\,|s\rangle\,, \tag{F14}$$

where $X(b)$ denotes the tensor-product unitary applying X^{b_i} to qubit i.The above is equivalent to

$$0 = \prod_{j:\ v_j = 0} \langle [(1\!:\!a)F]_j | s_j\rangle\,. \tag{F15}$$

We therefore have at least one j such that $v_j = 0$. We first consider the case where there is exactly one j such that $v_j = 0$. Then, for this j, we have $\forall a \in \{0,1\}^n$

$$\langle [(1\!:\!a)F]_j | s_j\rangle = 0. \tag{F16}$$

Choosing $a = 00\ldots0$ and computing the matrix multiplication $(1\!:\!a)F$ we obtain

$$F_{1j} \neq s_j. \tag{F17}$$

Now choosing (for each k individually) $a_k = \delta_{kj}$, we obtain

$$F_{1j} + F_{kj} \neq s_j. \tag{F18}$$

This implies that for $k > 1$ we have $F_{kj} = 0$, and since the column $F_{:,j}$ cannot consist of entirely zeros as F is invertible (indeed the inverse of F is the transpose of G), we have

$$F_{1j} = 1 \implies s_j = 0. \tag{F19}$$

We are therefore in case 2. We note that the assumption that exactly one of the v_j is equal to zero is necessary in the above, to allow us to change the bitstring a without the j in Eq. (F16) changing.

Finally we consider the case where there exist distinct j, k such that $v_j = v_k = 0$. If either of s_j or s_k equals 0 we are in case 2, otherwise both are equal to 1 and we are in case 3. □

In what follows we will neglect case 1 since if $\omega = 0$ the state is independent of the rest of the CH-form and all computations are trivial. We now provide two lemmas that show that any CH-form for a tensor-product state $|0\rangle \otimes |\sigma\rangle$ may be efficiently brought into a convenient form.

Lemma 16. *Given $\omega \neq 0$ and*

$$|0\rangle \otimes |\sigma\rangle = \omega U_C U_H\,|s\rangle\,, \tag{F20}$$

where the CH-form on the right is given by $\mathcal{F} = \{F, G, M, \gamma, v, s, \omega\}$, we can compute $\mathcal{F}' = \{F', G', M', \gamma', v', s', \omega'\}$ defining the same state such that there is at most one index j with $v'_j = 0$, $s'_j = 1$.

Proof. Assume there are multiple indices j such that $v_j = 0$ and $s_j = 1$. We recall that the controlled X gate, $\mathrm{CX_{p,q}}$, is its own inverse, so we have

$$\omega U_C U_H |s\rangle = \omega U_C \mathrm{CX_{p,q}} \mathrm{CX_{p,q}} U_H |s\rangle , \tag{F21}$$

for all $p \neq q$. Let a be the least index such that $v_a = 0$ and $s_a = 1$. Then, for all $b > a$ such that $v_b = 0$ and $s_b = 1$, we insert a pair of controlled X gates controlled on a and targeted on b. Since $\mathrm{CX_{a,b}}$ is its own inverse, this insertion does not change the quantum state we are representing. We then let the left hand $\mathrm{CX_{a,b}}$ act on U_C in accordance with the update rules given in Eqs.(E4a)-(E4c), while the right hand $\mathrm{CX_{a,b}}$ acts on the state $U_H |s\rangle$. Since $v_a = v_b = 0$ and $s_a = s_b = 1$, the action of this is to flip s_b to 0. □

Lemma 17. *Consider $\omega \neq 0$ and*

$$|0\rangle \otimes |\sigma\rangle = \omega U_C U_H |s\rangle , \tag{F22}$$

where the CH-form on the right is given by $\mathcal{F} = \{F, G, M, \gamma, v, s, \omega\}$, and assume there is at most one j such that $v_j = 0$, and $s_j = 1$. Then, the first row of G is non-zero only for elements G_{1p} for which $s_p = v_p = 0$.

Proof. First assume there is no j such that $v_j = 0$ while $s_j = 1$. Let p be an index such that $G_{1p} = 1$ and let $x = e_p G^T$, where e_p is the vector which has 1 in the p^{th} entry and 0 in all other entries. We consider the inner product of $\langle x| (|0\rangle \otimes |\sigma\rangle)$. Since $G_{1p} = 1$, we have that $x_0 = 1$, so the inner product equals 0. From Eq. (E5) we read

$$0 = \prod_{j:\ v_j=0} \langle (xF)_j |0\rangle = \prod_{j:\ v_j=0} \langle (e_p G^T F)_j |0\rangle = \prod_{j:\ v_j=0} \langle (e_p)_j |0\rangle \implies v_p = 0, \tag{F23}$$

since $G^T F$ is the identity matrix and $(e_p)_j = \delta_{pj}$.

Now assume there exists a single index k such that $v_k = 0$, $s_k = 1$. Consider the inner product

$$\langle e_k G^T | (|0\rangle \otimes |\sigma\rangle) = a \prod_{j:\ v_j=0} \langle (e_k G^T F)_j | s_j \rangle = a \prod_{j:\ v_j=0} \langle (e_k)_j | s_j \rangle = a \prod_{j\neq k:\ v_j=0} \langle (e_p)_j |0\rangle \cdot \langle (e_k)_k |1\rangle = a, \tag{F24}$$

where $a \neq 0$ is a constant given by Eq. (E5). Since $a \neq 0$ we have $\langle e_k G^T | |0\rangle |\sigma\rangle \neq 0$, which implies $(e_k G^T)_1 = 0$, and therefore $G_{1k} = 0$.

Finally, for a $p \neq k$ such that $G_{1p} = 1$ consider $x = (e_p + e_k) G^T$. Since $G_{1k} = 0$ and $G_{1p} = 1$, we have $x_1 = 1$ and hence

$$\begin{aligned} 0 = \langle x| (|0\rangle \otimes |\sigma\rangle) &= \prod_{j:\ v_j=0} \langle ((e_k + e_p) G^T F)_j | s_j \rangle = \prod_{j:\ v_j=0} \langle (e_k + e_p)_j | s_j \rangle \\ &= \prod_{j\neq k:\ v_j=0} \langle (e_p + e_k)_j |0\rangle \cdot \langle (e_p + e_k)_k |1\rangle = \prod_{j\neq k:\ v_j=0} \langle (e_p)_j |0\rangle \implies v_p = 0. \end{aligned} \tag{F25}$$

□

We now have all the ingredients to present the last part of the proof. We first ensure that $G_{11} = 1$. If this not the case, we choose a q such that $G_{1q} = 1$ (such a q exists since G is invertible) and insert a pair of swap gates using the identity

$$\omega U_C U_H |s\rangle = \omega U_C \mathrm{SWAP_{1,q}} \mathrm{SWAP_{1,q}} U_H |s\rangle , \tag{F26}$$

multiply the left hand $SWAP$ onto U_C (where it swaps the first and q^{th} column of G), and apply the right hand one to $U_H |s\rangle$ (where it swaps the first bit of v with the q^{th} bit of v and the first bit of s with the q^{th} bit of s).

The formula $G^T F = I$ implies that (the sums below are mod 2)

$$\sum_p G_{1p} F_{:,p} = \sum_{p: G_{1p}=1} F_{:,p} = e_1^T, \tag{F27}$$

since e_1^T is the first column of the identity matrix. We now consider all the indices $p > 1$ such that $G_{1p} = 1.$, Lemma 17 implies that for such a p the equation

$$\omega U_C U_H |s\rangle = \omega U_C \mathrm{CX_{p,1}} U_H |s\rangle \tag{F28}$$

holds, since $v_p = s_p = 0$ implies the p^{th} qubit of $U_H |s\rangle$ is in the state $|0\rangle$. Right-multiplying this $\text{CX}_{\text{p},1}$ onto U_C causes the p^{th} column of the F matrix to be added onto the 1^{st} column. The right-multiplication does not alter the first row of the G matrix. We therefore have the identity

$$\omega U_C U_H |s\rangle = \omega U_C \prod_{p:G_{1p}=1} \text{CX}_{\text{p},1} U_H |s\rangle , \tag{F29}$$

resulting in the update to the F matrix

$$F_{:,1} \leftarrow \bigoplus_{p:G_{1p}=1} F_{:,p} = e_1^T. \tag{F30}$$

Appendix G: Optimizing magic state count

By analysing the stabilisers we obtain it is possible to reduce the T gate count of a circuit. The size of the improvement depends on the circuit in question, but for some circuits this optimisation can have a significant impact.

We recall equation (33b), which expresses the probability of interest as the expectation value of a product Stabilizer state projectors

$$p = 2^{v-w} \langle T_{\boldsymbol{\phi}}^\dagger | \prod_{i=1}^{t-r} (I + g_i) | T_{\boldsymbol{\phi}}^\dagger \rangle, \tag{G1}$$

where the g_i are an independent set of t qubit Pauli operators. We can multiply the generators between themselves to obtain equivalent generating sets, and use this freedom to put the stabiliser table in $ZX(j)$ form for any qubit number j.

Recall that in $ZX(j)$ form at most two of the generators are non-identity on qubit j and either

1. All generators are identity on qubit j
2. A single generators is non identity on qubit j that generators's action on qubit j may be
   (a) X_j
   (b) Y_j
   (c) Z_j
3. Two generators are non-identity on qubit j, one has action X_j on qubit j and one has action Z_j.

We focus on the case where in $ZX(j)$ form there is a single stabiliser generator g_k which acts non-trivially on qubit j and that generator has action Z_j on qubit j. Of course this generator may also act non-trivially on other qubits. We rewrite equation (G1)

$$p = 2^{v-w} \langle T_{\boldsymbol{\phi}}^\dagger | (I + g_k) \prod_{i \neq k} (I + g_i) | T_{\boldsymbol{\phi}}^\dagger \rangle, \tag{G2}$$

since $|g_k|_j = Z$, for all $i \neq k$ $|g_i|_j = I$ and each of the states in the tensor product $|T_{\boldsymbol{\phi}}^\dagger\rangle$ are equatorial we have

$$\langle T_{\boldsymbol{\phi}}^\dagger | g_k \prod_{i \neq k}^{t-r} (I + g_i) | T_{\boldsymbol{\phi}}^\dagger \rangle = \langle T_{\phi_j}^\dagger | g_k | T_{\phi_j}^\dagger \rangle \tag{G3}$$

$$= 0 \tag{G4}$$

and therefore

$$p = 2^{v-w} \langle T_{\boldsymbol{\phi}}^\dagger | \prod_{i \neq k} (I + g_i) | T_{\boldsymbol{\phi}}^\dagger \rangle. \tag{G5}$$

We may therefore remove generator k from our set of generators without changing p. All remaining generators have $|g_i|_j = I$, and since products of generators happen component-wise this column of identities will remain in the stabiliser tableau regardless of subsequent changes in the generating set.

We therefore have a simple algorithm to reduce the number of generators needed in our generating set.

Algorithm 4 MAGIC-CONSTRAIN Given a stabilizer generating set $\langle G\rangle$ returns a stabilizer generating set $\langle G'\rangle$ such that $|G'| \leq |G|$ $\langle T_{\boldsymbol{\phi}}^{\dagger}| \prod_{g\in G'}(I+g)|T_{\boldsymbol{\phi}}^{\dagger}\rangle = \langle T_{\boldsymbol{\phi}}^{\dagger}| \prod_{g\in G}(I+g)|T_{\boldsymbol{\phi}}^{\dagger}\rangle$.

Input: A stabilizer generating set $G = \{g_1, \ldots, g_k\} \in \mathcal{G}(t,k)$.
Output: A stabilizer generating set $G = \{g_1, \ldots, g_k\} \in \mathcal{G}(t,k)$.

count $\leftarrow 0$
while count $< t - r$ **do**
  count $\leftarrow$ count $+1$
  $j \leftarrow 1$
  while $j \leq t - r$ **do**
    $G \leftarrow$ ZX-FORM(G, j)
    if G has a leading Z generator and no leading X generator **then**
      delete the leading Z generator.
      $r \leftarrow r + 1$
    end if
    $j \leftarrow j + 1$
  end while
end while

We note that the result of applying algorithm 4 is a set of stabiliser generators with some number $0 \leq q$ qubits for which the action of every generator on that qubit is the identity. Examining equation (G1) again we see that if qubit k has the property that every stabiliser generator g satisfies the equation $|g|_k = I$ then this qubit contributes an overall factor of

$$\langle T_{\phi_k}^{\dagger}|I|T_{\phi_k}^{\dagger}\rangle = 1 \tag{G6}$$

to the probability p. We may therefore delete this qubit from our stabiliser tableau without changing p.

The impact of imposing the region c constraints is therefore to reduce the stabiliser generator count by from $t - r$ to $t - r - q$, while simultaneously reducing the qubit count by q so r is unchanged. If in addition there are qubits on which every stabiliser generator is the identity arising from the region a and b constraints we can also delete these, reducing $t - r$ but increasing r. The impact of this change is less clear since the run-time of RAWESTIM has components which scale in $t - r$ and in r but we expect it to be a net positive.

[1] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).
[2] A. W. Harrow and A. Montanaro, Quantum computational supremacy, Nature **549**, 203 (2017).
[3] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).
[4] H.-S. Zhong *et al.*, Quantum computational advantage using photons, Science **370**, 1460 (2020).
[5] S. T. Flammia and Y.-K. Liu, Direct fidelity estimation from few Pauli measurements, Phys. Rev. Lett. **106**, 230501 (2011).
[6] M. Kliesch, Lecture notes: Characterization, verification, and validation of quantum systems, (2019).
[7] V. Havlí**c**ek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature **567**, 209 (2019).
[8] A. Fatima and I. L. Markov, Faster Schrödinger-style simulation of quantum circuits, arXiv:2008.00216 (2020).
[9] H. De Raedt, F. Jin, D. Willsch, M. Willsch, N. Yoshioka, N. Ito, S. Yuan, and K. Michielsen, Massively parallel quantum computer simulator, eleven years later, Comput. Phys. Commun. **237**, 47 (2019).
[10] I. Markov and Y. Shi, Simulating quantum computation by contracting tensor networks, SIAM J. Comput. **38**, 963 (2008).
[11] K. De Raedt, K. Michielsen, H. De Raedt, B. Trieu, G. Arnold, M. Richter, T. Lippert, H. Watanabe, and N. Ito, Massively parallel quantum computer simulator, Comput. Phys. Commun. **176**, 121 (2007).
[12] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, Quantum supremacy is both closer and farther than it appears, arXiv:1807.10749 (2018).

[13] D. Gottesman, The Heisenberg representation of quantum computers, arXiv quant-ph/9807006 (1998).
[14] S. D. Bartlett, B. C. Sanders, S. L. Braunstein, and K. Nemoto, Efficient classical simulation of continuous variable quantum information processes, Phys. Rev. Lett. **88**, 097904 (2002).
[15] B. M. Terhal and D. P. DiVincenzo, Classical simulation of noninteracting-fermion quantum circuits, Phys. Rev. A **65**, 032325 (2002).
[16] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).
[17] R. Jozsa and A. Miyake, Matchgates and classical simulation of quantum circuits, Proc. R. Soc. A **464**, 3089 (2008).
[18] P. Rall, D. Liang, J. Cook, and W. Kretschmer, Simulation of qubit quantum circuits via Pauli propagation, Phys. Rev. A **99**, 062337 (2019).
[19] H. Pashayan, S. D. Bartlett, and D. Gross, From estimation of quantum probabilities to simulation of quantum circuits, Quantum **4**, 223 (2020).
[20] H. Pashayan, *On the classical simulability of quantum circuits*, Ph.D. thesis, University of Sydney (2019).
[21] M. Howard and E. Campbell, Application of a resource theory for magic states to fault-tolerant quantum computing, Phys. Rev. Lett. **118**, 090501 (2017).
[22] J. R. Seddon, B. Regula, H. Pashayan, Y. Ouyang, and E. T. Campbell, Quantifying quantum speedups: improved classical simulation from tighter magic monotones, arXiv:2002.06181 (2020).
[23] H. Pashayan, J. J. Wallman, and S. D. Bartlett, Estimating outcome probabilities of quantum circuits using quasiprobabilities, Phys. Rev. Lett. **115**, 070501 (2015).
[24] V. Veitch, C. Ferrie, D. Gross, and J. Emerson, Negative quasi-probability as a resource for quantum computation, New J. Phys. **14**, 113011 (2012).
[25] A. Mari and J. Eisert, Positive Wigner functions render classical simulation of quantum computation efficient, Phys. Rev. Lett. **109**, 230503 (2012).
[26] H. J. Garcia, I. L. Markov, and A. W. Cross, Efficient inner-product algorithm for stabilizer states, arXiv:1210.6646 (2012).
[27] H. J. García, I. L. Markov, and A. W. Cross, On the geometry of stabilizer states, Quant. Inf. and Comp. **14**, 683 (2014).
[28] S. Bravyi, G. Smith, and J. A. Smolin, Trading classical and quantum computational resources, Phys. Rev. X **6**, 021043 (2016).
[29] S. Bravyi and D. Gosset, Improved classical simulation of quantum circuits dominated by Clifford gates, Phys. Rev. Lett. **116**, 250501 (2016).
[30] S. Bravyi, D. Browne, P. Calpin, E. Campbell, D. Gosset, and M. Howard, Simulation of quantum circuits by low-rank stabilizer decompositions, Quantum **3**, 181 (2019).
[31] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, On universal and fault-tolerant quantum computing: a novel basis and a new constructive proof of universality for Shor's basis, in *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)* (1999) pp. 486–494.
[32] C. Ferrie and J. Emerson, Frame representations of quantum mechanics and the necessity of negativity in quasi-probability representations, J. Phys. A **41**, 352001 (2008).
[33] D. Gross, Hudson's theorem for finite-dimensional quantum systems, J. Math. Phys. **47**, 122107 (2006).
[34] K. Gibbons, M. Hoffman, and W. Wootters, Discrete phase space based on finite fields, Phys. Rev. A **70** (2004).
[35] O. Reardon-Smith, *Clifford-T-estimator* (2020).
[36] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. Cabrera-Hernández, J. Carballo-Franquis, A. Chen, C. Chen, *et al.*, Qiskit: An open-source framework for quantum computing, (2019).
[37] G. Duclos-Cianci and D. Poulin, Reducing the quantum-computing overhead with complex gate distillation, Phys. Rev. A **91**, 042315 (2015).
[38] D. Gottesman and I. L. Chuang, Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations, Nature **402**, 390 (1999).
[39] X. Zhou, D. W. Leung, and I. L. Chuang, Methodology for quantum logic gate construction, Phys. Rev. A **62**, 052316 (2000).
[40] T. P. Hayes, A large-deviation inequality for vector-valued martingales, Combinatorics, Probability and Computing (2005).