Improved classical simulation of quantum circuits dominated by Clifford gates

Sergey Bravyi[1] and David Gosset[2]

[1] *IBM T.J. Watson Research Center, Yorktown Heights NY 10598*
[2] *Walter Burke Institute for Theoretical Physics and Institute for Quantum Information and Matter, California Institute of Technology*

(Dated: January 31, 2017)

The Gottesman-Knill theorem asserts that a quantum circuit composed of Clifford gates can be efficiently simulated on a classical computer. Here we revisit this theorem and extend it to quantum circuits composed of Clifford and T gates, where T is the single-qubit 45° phase shift. We assume that the circuit outputs a bit string x obtained by measuring some subset of w qubits. Two simulation tasks are considered: (1) computing the probability of a given output x, and (2) sampling x from the output probability distribution. It is shown that these tasks can be solved on a classical computer in time $poly(n,m) + 2^{0.5t}t^3$ and $poly(n,m) + 2^{0.23t}t^3w^3$ respectively, where t is the number of T-gates, m is the total number of gates, and n is the number of qubits. The proposed simulation algorithms may serve as a verification tool for medium-size quantum computations that are dominated by Clifford gates. The main ingredient of both algorithms is a subroutine for approximating the norm of an n-qubit state which is given as a linear combination of χ stabilizer states. The subroutine runs in time $\chi n^3\epsilon^{-2}$, where ϵ is the relative error. We also develop techniques for approximating tensor products of "magic states" by linear combinations of stabilizer states. To demonstrate the power of the new simulation methods, we performed a classical simulation of a hidden shift quantum algorithm with 40 qubits, a few hundred Clifford gates, and nearly 50 T-gates.

I. INTRODUCTION

The path towards building a large-scale quantum computer will inevitably require verification and validation of small quantum devices. One way to check that such a device is working properly is to simulate it on a classical computer. This becomes impractical at some point because the cost of classical simulation typically grows exponentially with the size of a quantum system. With this fundamental limitation in mind it is natural to ask how well we can do in practice.

Simulation methods which store a complete description of an n-qubit quantum state as a complex vector of size 2^n are limited to a small number of qubits $n \approx 30$. For example, a state-of-the art implementation has been used to simulate Shor's factoring algorithm with 31 qubits and roughly half a million gates [1]. For certain restricted classes of quantum circuits it is possible to do much better [2–6]. Most significantly, the Gottesman-Knill theorem allows efficient classical simulation of quantum circuits composed of gates in the so-called Clifford group [2]. In practice this allows one to simulate such circuits with thousands of qubits [1, 3]. It also means that a quantum computer will need to use gates outside of the Clifford group in order to achieve useful speedups over classical computation. The full power of quantum computation can be recovered by adding a single non-Clifford gate to the Clifford group. A simple choice is the single-qubit $T = |0\rangle\langle 0| + e^{i\pi/4}|1\rangle\langle 1|$ gate; the Clifford+T gate set obtained in this way is a natural instruction set for small-scale fault-tolerant quantum computers based on the surface code [7, 8], and has been at the centre of a recent renaissance in classical techniques for compiling quantum circuits [9–11].

In this paper we present two new algorithms for classical simulation of quantum circuits over the Clifford+T gate set. The runtime of the algorithms is polynomial in the number of qubits and the number of Clifford gates in the circuit but exponential in the number of T gates, or T-count. This exponential scaling is sufficiently mild that we anticipate a classical simulation of Clifford+T circuits with a few hundred qubits and T-count $t \leq 50$ can be performed on a medium-size computer cluster. Thus our algorithms may serve as a verification tool for small quantum computations dominated by Clifford gates. Such computations arise naturally if a logical quantum circuit is realized fault-tolerantly using some stabilizer code. The first demonstrations of logical quantum circuits using the surface code are likely to be dominated by Clifford gates due to a high implementation cost associated with logical T-gates [12, 13].

To describe our results let us fix some notation. A Clifford+T quantum circuit of length m acting on n qubits is a unitary operator $U = U_m \cdots U_2 U_1$, where each U_j is a one- or two-qubit gate from the set $\{H, S, T, CNOT\}$ where H is the Hadamard gate and $S = |0\rangle\langle 0| + i|1\rangle\langle 1|$. We shall write $m = c + t$, where c is the number of Clifford gates $(H, S, CNOT)$ and t is the number of T-gates also known as the T-count. Applying U to the initial state $|0^n\rangle$ and measuring some fixed output register $Q_{out} \subseteq [n]$ in the $0,1$-basis generates a random bit string x of length $w = |Q_{out}|$. A string x appears with probability

$$P_{out}(x) = \langle 0^n|U^\dagger \Pi(x) U|0^n\rangle, \tag{1}$$

where $\Pi(x)$ projects Q_{out} onto the basis state $|x\rangle$ and acts trivially on the remaining qubits.

Our first result is a classical Monte Carlo algorithm

that approximates the probability $P_{out}(x)$ for a given string $x \in \{0,1\}^w$ with a specified relative error ϵ and a failure probability p_f. The algorithm has runtime

$$\tau = O\left((w+t)(c+t) + (n+t)^3 + 2^{\beta t} t^3 \epsilon^{-2} \log(p_f^{-1})\right), \quad (2)$$

where $\beta \leq 1/2$ is a constant that depends on the implementation details. For example, assuming that ϵ and p_f are some fixed constants and $w \leq t \leq n \leq c$, the runtime becomes

$$\tau = O(n^3 + ct + 2^{\beta t} t^3).$$

Our second result is a classical algorithm that allows one to sample the output string x from a distribution which is ϵ-close to P_{out} with respect to the L_1-norm. The sampling algorithm has runtime

$$\tau = \tilde{O}\left(w(w+t)(c+t) + w(n+t)^3 + 2^{\gamma t} t^3 w^3 \epsilon^{-4}\right), \quad (3)$$

where the $\tilde{O}$ notation hides a factor logarithmic in w and ϵ^{-1}, and

$$\gamma \leq -2 \log_2(\cos(\pi/8)) \approx 0.228 \quad (4)$$

is a constant that depends on the implementation details. We expect the sampling algorithm to be practical when w is small and ϵ is not too small. For example, assuming that the circuit outputs a single bit ($w = 1$), ϵ is a fixed constant, and $t \leq n \leq c$, the runtime becomes

$$\tau = O(n^3 + ct + 2^{\gamma t} t^3).$$

Both algorithms can be divided into independent subroutines with a runtime $O(t^3)$ each and thus support a large amount of parallelism. We provide pseudocode for the main subroutines used in the algorithms and a timing analysis for the MATLAB implementation [14] in the Supplemental Material.

Since the simulation runtime is likely to be dominated by the terms exponential in t, one may wish to minimize the exponents β, γ in Eqs. (2,3). These exponents are related to the stabilizer rank [15] of a magic state

$$|A\rangle = 2^{-1/2}(|0\rangle + e^{i\pi/4}|1\rangle).$$

Recall that a t-qubit state is called a stabilizer state if it has the form $V|0^t\rangle$, where V is a quantum circuit composed of Clifford gates. Stabilizer states form an overcomplete basis in the Hilbert space of t qubits. Let $\chi_t(\delta)$ be the smallest integer χ such that $A^{\otimes t}$ can be approximated with an error at most δ by a linear combination of χ stabilizer states (here the approximating state ψ should satisfy $|\langle A^{\otimes t}|\psi\rangle|^2 \geq 1-\delta$). The runtime scaling in Eq. (2) holds for any exponent β such that $\chi_t(0) = O(2^{\beta t})$ for all sufficiently large t. Using the results of [15] one can choose $\beta = (1/6)\log_2(7) \approx 0.47$. Stronger upper bounds on the stabilizer rank $\chi_t(0)$ could improve the runtime scaling in Eq. (2). Likewise, the runtime scaling in Eq. (3) holds for any exponent γ such that $\chi_t(\delta) = O(2^{\gamma t})$ for any constant $\delta > 0$ and all sufficiently large t. For simplicity here we assumed that the precision parameter ϵ in Eq. (3) is a constant. In this paper we propose a systematic method of finding approximate stabilizer decompositions of $A^{\otimes t}$ which yields an upper bound $\chi_t(\delta) = O(2^{\gamma t}\delta^{-1})$, where $\gamma \approx 0.228$, see Eq. (4). We conjecture that this upper bound is tight.

We implemented our classical sampling algorithm in MATLAB and used it to simulate a class of benchmark quantum circuits on $n = 40$ qubits, with a few hundred Clifford gates, and T-count $t \leq 48$. Specifically, we simulated a quantum algorithm which solves the hidden shift problem for non-linear Boolean functions [16]. An instance of the hidden shift problem is defined by a pair of oracle functions $f, f' : \mathbb{F}_2^n \to \{\pm 1\}$ and a hidden shift string $s \in \mathbb{F}_2^n$. It is promised that f is a bent (maximally non-linear) function, that is, the Hadamard transform of f takes values ± 1. It is also promised that f' is the shifted version of the Hadamard transform of f, that is,

$$f'(x \oplus s) = 2^{-n/2} \sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} f(y) \quad \text{for all } x \in \mathbb{F}_2^n. \quad (5)$$

Here $\oplus$ stands for the bit-wise XOR. The goal is to learn the hidden shift s by making as few queries to f and f' as possible. The classical query complexity of this problem is known to be linear in n, see Theorem 8 of Ref. [16]. In the quantum setting, f and f' are given as diagonal n-qubit unitary operators O_f and $O_{f'}$ such that $O_f|x\rangle = f(x)|x\rangle$ and $O_{f'}|x\rangle = f'(x)|x\rangle$ for all $x \in \mathbb{F}_2^n$. A quantum algorithm can learn s by making a single query to each of these oracles, as can be seen from the identity [16]

$$|s\rangle = U|0^n\rangle, \quad U \equiv H^{\otimes n} O_{f'} H^{\otimes n} O_f H^{\otimes n}. \quad (6)$$

This hidden shift problem is ideally suited for our benchmarking task for two reasons. First, the algorithm produces a deterministic output, i.e., the output is a computational basis state $|s\rangle$ for some n-bit string s. Because of this we achieve the most favorable runtime scaling in Eq. (3) since each bit of s can be learned by calling the sampling algorithm with a single-qubit output register ($w = 1$) and a constant statistical error ϵ. Second, the T-count of the algorithm can be easily controlled by choosing a suitable bent function. Indeed, the non-oracle part of the algorithm consists only of Hadamard gates. We show that for a large class of bent functions f (from the so-called Maiorana-McFarland family) the oracles O_f and $O_{f'}$ can be constructed using Clifford gates and only a few T gates, see the Supplemental Material for details.

The numerical simulations were performed for two randomly generated instances of the hidden shift problem with $n = 40$ qubits. For each of these instances we simulated the quantum circuit for the hidden shift algorithm, i.e., the circuit implementing the unitary U described above. The T-counts of the two simulated circuits are $t = 40$ and $t = 48$ respectively. Since the hidden shift s is known beforehand, we are able to verify correctness

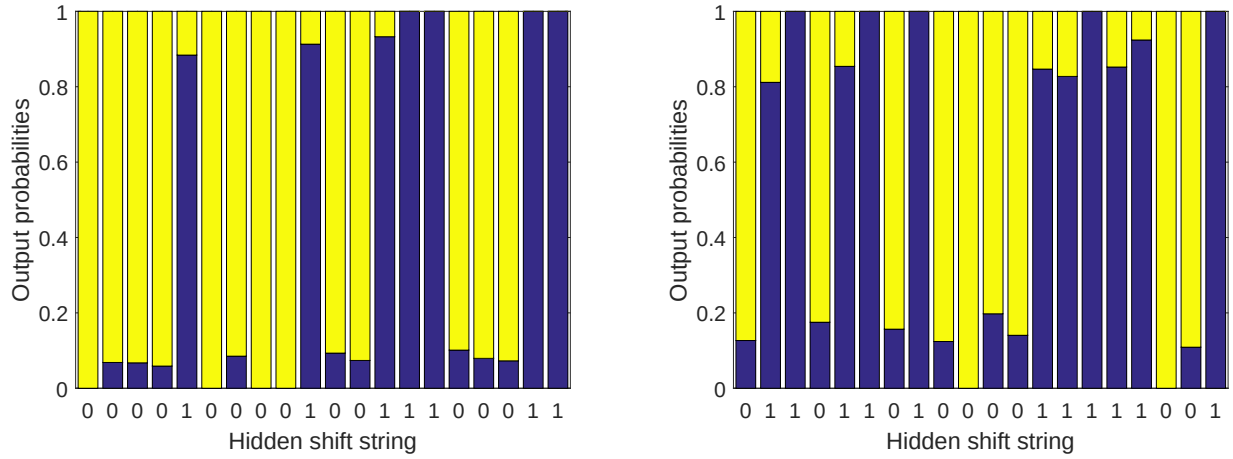

FIG. 1. Output single-qubit probability distributions obtained by a classical simulation of the hidden shift quantum algorithm on $n = 40$ qubits. Only one half of all qubits are shown (qubits $21, 22, \ldots, 40$). The final state of the algorithm is $|s\rangle = U|0^n\rangle$, where s is the hidden shift string to be found and U is a Clifford+T circuit with the T-count $t = 40$ (left) and $t = 48$ (right). In both cases the circuit U contains a few hundred Clifford gates. For each qubit the probability of measuring '1' in the final state is indicated in blue. The x-axis labels indicate the correct hidden shift bits. The entire simulation took several hours on a laptop computer.

of the simulation. Our results are presented in Fig. 1. As one can see from the plots, the output probability distribution of each qubit has most of its weight at the corresponding value of the hidden shift bit. Only the output probabilities for qubits $21, 22, \ldots, 40$ are shown because our algorithm perfectly recovered the first half of the hidden shift bits $1, 2, \ldots, 20$. This perfect recovery occurs due to the special structure of the chosen bent functions, see the Supplemental Material for further details.

The rest of the paper is organized as follows. In Section II we give an overview of our main techniques. In Section III we summarize some basic facts concerning stabilizer states. We present our classical simulation algorithms for Clifford+T circuits in Section IV. Finally, we show how to approximate tensor products of magic states by linear combinations of stabilizer states in Section V. In the Supplemental Material we provide pseudocode for the main subroutines used in our algorithms, and we discuss further details of the simulations reported in Fig. 1.

II. SKETCH OF TECHNIQUES

Following Ref. [15], we simulate a Clifford+T circuit classically using three basic steps. First, each T-gate in the original circuit is replaced by a certain gadget that contains only Clifford gates and a $0, 1$-measurement. The Clifford gates may be classically controlled by the measurement outcome. The gadget consumes one copy of the magic state $|A\rangle$. This gives an equivalent 'gadgetized' circuit acting on a non-stabilizer initial state that contains t copies of $|A\rangle$. We show how to remove all intermediate measurements from the gadgetized circuit by replacing the outcomes of these measurements by random uniform postselection bits. Accordingly, we replace the classically controlled Clifford gates by a suitable random ensemble of uncontrolled Clifford gates. Second, the initial magic state $A^{\otimes t}$ is represented (exactly or approximately) as a linear combination of $\chi \ll 2^n$ stabilizer states. The action of the gadgetized circuit on each term in this linear combination can be efficiently simulated using the standard Gottesman-Knill theorem since the gadgetized circuit contains only Clifford gates. This allows us to represent the final state before the measurement of Q_{out} as a linear combination of χ stabilizer states. We simulate the measurement of Q_{out} on this final state independently for each term in the linear combination (we also have to simulate certain additional post-selective measurements introduced at the first step). This is possible due to the fact that $0, 1$-measurements map stabilizer states to stabilizer states. The final post-measurement state is a linear combination of at most χ stabilizer states. The third and the most time consuming step is computing the norm of the post-measurement state. This norm is simply related to the quantity of interest, such as the output probability $P_{out}(x)$. We show how to obtain a square-root speedup in this step compared with Ref. [15] reducing the runtime scaling from χ^2 to χ. This is achieved using a novel subroutine for approximating the norm of a linear combination of stabilizer states. The subroutine has runtime $O(\chi t^3 \epsilon^{-2})$, where χ is the number of terms in the linear combination, t is the number of qubits, and ϵ is the relative error. We expect that this subroutine may find applications in other contexts. We achieve a further speedup compared with Ref. [15] by reducing the scaling $\chi \approx 2^{0.47t}$ to $\chi \approx 2^{0.23t}$ by developing techniques for approximate stabilizer decompositions of $A^{\otimes t}$. Although in general the simulation algorithm based on approximate

stabilizer decomposition cannot accurately compute individual probabilities of the output distribution, we show that it can be used for sampling from the output distribution with a small statistical error.

III. STABILIZER FORMALISM

Before moving further, let us state some facts concerning stabilizer groups and stabilizer states. Let $\mathcal{P}_n$ be the n-qubit Pauli group. Any element of $\mathcal{P}_n$ has the form $i^m P_1 \otimes \cdots \otimes P_n$, where each factor P_a is either the identity or a single-qubit Pauli operator X, Y, Z and $m \in \mathbb{Z}_4$. An abelian subgroup $\mathcal{G} \subseteq \mathcal{P}_n$ is called a stabilizer group if $-I \notin \mathcal{G}$. Each stabilizer group has the form $\mathcal{G} = \langle G_1, \ldots, G_r \rangle$ for some generating set of pairwise commuting self-adjoint Pauli operators $G_1, \ldots, G_r \in \mathcal{G}$ such that $|\mathcal{G}| = 2^r$. The integer r is called the dimension of $\mathcal{G}$ and is denoted $r = \dim(\mathcal{G})$. A state ψ is said to be stabilized by $\mathcal{G}$ if $P\psi = \psi$ for all $P \in \mathcal{G}$. States stabilized by $\mathcal{G}$ span a "codespace" of dimension 2^{n-r}. A projector onto a codespace has the form

$$\Pi_{\mathcal{G}} = 2^{-r} \sum_{P \in \mathcal{G}} P. \tag{7}$$

A pure n-qubit state ψ is a stabilizer state iff $|\psi\rangle = U|0^n\rangle$ for some Clifford unitary U. Any stabilizer state ψ is uniquely defined (up to the overall phase) by a stabilizer group $\mathcal{G} \subseteq \mathcal{P}_n$ of dimension n such that ψ is the only state stabilized by $\mathcal{G}$. Let $\mathcal{S}_n$ be the set of all n-qubit stabilizer states. This set is known to be a 2-design [17], that is,

$$|\mathcal{S}_n|^{-1} \sum_{\psi \in \mathcal{S}_n} |\psi\rangle\langle\psi|^{\otimes 2} = \int d\mu(\phi) |\phi\rangle\langle\phi|^{\otimes 2}, \tag{8}$$

where the integral is with respect to the Haar measure on the set of all normalized n-qubit states ϕ.

Throughout the paper we assume that stabilizer states are represented in a certain standard form defined in Appendix B. In this representation, three basic tasks can be performed efficiently. First, one can compute the inner product between stabilizer states [3, 15, 18]. More precisely, consider stabilizer states $\psi, \phi \in \mathcal{S}_n$. Then $\langle\psi|\phi\rangle = b 2^{-p/2} e^{i\pi m/4}$ for some $b = 0, 1$, integer $p \in [0, n]$ and $m \in \mathbb{Z}_8$ that can be computed in time $O(n^3)$, see Ref. [15]. Pseudocode for computing the inner product $\langle\psi|\phi\rangle$ can be found in Appendix C. Secondly, a projection of any stabilizer state onto the codespace of any stabilizer code is a stabilizer state which is easy to compute. More precisely, suppose $\mathcal{G} \subseteq \mathcal{P}_n$ is a stabilizer group and $\varphi \in \mathcal{S}_n$. Then $\Pi_{\mathcal{G}}|\varphi\rangle = b 2^{-p/2} |\phi\rangle$ for some $b = 0, 1$, some integer $p \geq 0$, and stabilizer state $\phi \in \mathcal{S}_n$. One can compute b, p, ϕ in time $O(rn^2)$ as explained in Appendix E. Recall that $r = \dim(\mathcal{G})$. Finally, one can generate a random stabilizer state drawn from the uniform distribution on $\mathcal{S}_n$ in time $O(n^2)$, see Appendix D.

IV. CLASSICAL SIMULATION ALGORITHMS

First consider the task of approximating the output probability $P_{out}(x)$. The algorithm described below consists of two stages with runtimes

$$\tau_1 = O\left((w+t)(c+t) + (n+t)^3\right)$$

and

$$\tau_2 = O(2^{\beta t} t^3 \epsilon^{-2} \log(p_f^{-1})).$$

The first stage computes a stabilizer group $\mathcal{G} \subseteq \mathcal{P}_t$ and an integer u such that

$$P_{out}(x) = 2^{-u} \langle A^{\otimes t} | \Pi_{\mathcal{G}} | A^{\otimes t} \rangle. \tag{9}$$

We begin by replacing each T-gate in the original circuit U by the well-known gadget [19] shown in Fig. 2. The gadget implements the T-gate by performing Clifford gates CNOT, S, and a $0,1$-measurement. Each measurement outcome appears with the probability $1/2$. The gate S is applied only if the outcome is '1'. The gadget also consumes one copy of the magic state $|A\rangle$ which is destroyed in the process.

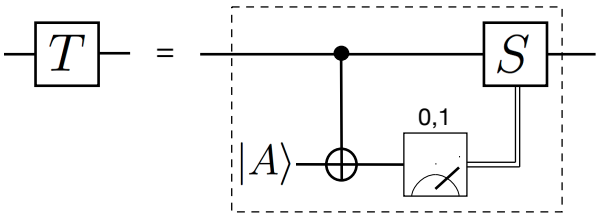

FIG. 2. The T-gate gadget. The Clifford gate S is classically controlled by the measurement outcome. Both outcomes appear with probability $1/2$.

Suppose we postselect the outcome '0' in each gadget, i.e. replace each measurement by a projector $|0\rangle\langle 0|$. This removes the classically controlled S-gates such that each gadget adds a single CNOT to the original circuit U. Let V be the modified version of U. By definition, V acts on $n+t$ qubits and contains $c+t$ Clifford gates. Let us agree that the t ancillary qubits initialized in the magic state are appended at the end of n computational qubits such that the circuit V acts on the initial state $|0^n A^{\otimes t}\rangle$. Combining the final measurement projector $\Pi(x) = |x\rangle\langle x|_{Q_{out}} \otimes I_{else}$ with the projectors $|0\rangle\langle 0|$ acting on the ancillary qubits gives a projector

$$\Pi = \Pi(x) \otimes |0^t\rangle\langle 0^t|$$

acting on $n+t$ qubits such that

$$P_{out}(x) = 2^t \langle 0^n A^{\otimes t} | V^\dagger \Pi V | 0^n A^{\otimes t} \rangle. \tag{10}$$

Here we noted that the postselection probability is 2^{-t}. Obviously, $\Pi = \Pi_{\mathcal{W}}$ for a stabilizer group $\mathcal{W} \subseteq \mathcal{P}_{n+t}$ of dimension $w+t$. Namely, let $q(j)$ be the j-th qubit

of Q_{out}. Generators of $\mathcal{W}$ are $R_j = (-1)^{x_j} Z_{q(j)}$ for $j = 1, \ldots, w$ and $R_{w+j} = Z_{n+j}$ for $j = 1, \ldots, t$. Since the conjugation by V maps Pauli operators to Pauli operators, we get $V^\dagger \Pi_{\mathcal{W}} V = \Pi_{\mathcal{V}}$, where $\mathcal{V}$ is a stabilizer group of dimension $w+t$ generated by $R'_j = V^\dagger R_j V \in \mathcal{P}_{n+t}$ with $j = 1, \ldots, w+t$. Assuming that the action of a single Clifford gate on a Pauli operator can be computed in time $O(1)$, one can compute each generator R'_j in time $O(c+t)$. Accordingly, $\mathcal{V}$ can be computed in time $O((w+t)(c+t))$.

Let $\mathcal{V}_0$ be the subgroup of $\mathcal{V}$ that includes all Pauli operators which act as I or Z on each of the first n qubits. Let $v = \dim(\mathcal{V}_0)$. A generating set $Q_1, \ldots, Q_v \in \mathcal{V}_0$ can be computed in time $O(n(w+t)+(w+t)^3) = O((n+t)^3)$ using standard linear algebra. We get

$$\langle 0^n | V^\dagger \Pi V | 0^n \rangle = \langle 0^n | \Pi_{\mathcal{V}} | 0^n \rangle = 2^{-w-t+v} \langle 0^n | \Pi_{\mathcal{V}_0} | 0^n \rangle. \tag{11}$$

since $\langle 0^n | P | 0^n \rangle = 0 \quad \forall\ P \in \mathcal{V} \setminus \mathcal{V}_0$. Define t-qubit Pauli operators $G_i = \langle 0^n | Q_i | 0^n \rangle$, $i = 1, \ldots, v$. These operators pairwise commute since $\mathcal{V}_0$ is abelian and Q_i commute with each other on the first n qubits. If $-I \in \langle G_1, \ldots, G_v \rangle$ then there exists $Q \in \mathcal{V}_0$ with $-I = \langle 0^n | Q | 0^n \rangle$ and therefore

$$\Pi_{\mathcal{V}} |0^n\rangle = \Pi_{\mathcal{V}} Q |0^n\rangle = -\Pi_{\mathcal{V}} |0^n\rangle = 0$$

in which case $P_{out}(x) = 0$ and we are done. Let us now consider the case $-I \notin \langle G_1, \ldots, G_v \rangle$. In this case let $\mathcal{G} \subseteq \mathcal{P}_t$ be the stabilizer group generated by $G_1, \ldots, G_v$ and $r = \dim(\mathcal{G})$. One can check the condition $-I \notin \langle G_1, \ldots, G_v \rangle$ and compute r in time $O(t^3)$. Without loss of generality, $\mathcal{G} = \langle G_1, \ldots, G_r \rangle$. Noting that $\mathcal{V}_0$ must contain 2^{v-r} elements acting trivially on the last t qubits yields $\langle 0^n | \Pi_{\mathcal{V}_0} | 0^n \rangle = \Pi_{\mathcal{G}}$. This proves Eq. (9) with $u = w - v$ and the stabilizer group $\mathcal{G}$ defined above. Combining all the steps needed to compute $\mathcal{G}$ gives the promised runtime $\tau_1 = O((w+t)(c+t) + (n+t)^3)$.

The second stage of the algorithm computes the expectation value in Eq. (9) by decomposing $|A^{\otimes t}\rangle$ into a linear combination of stabilizer states. Suppose

$$|A^{\otimes t}\rangle = \sum_{a=1}^{\chi} y_a |\varphi_a\rangle \tag{12}$$

for some stabilizer states $\varphi_a \in \mathcal{S}_t$ and some coefficients y_a. For each $a = 1, \ldots, \chi$ compute $b_a \in \{0, 1\}$, an integer $p_a \geq 0$ and a stabilizer state $\phi_a \in \mathcal{S}_t$ such that

$$\Pi_{\mathcal{G}} |\varphi_a\rangle = b_a 2^{-p_a/2} |\phi_a\rangle.$$

see Appendix E for details. As stated above, this computation takes time $O(\chi t^3)$. Introducing new coefficients $z_a = 2^{-(u+p_a)/2} y_a b_a$ and using Eqs. (9,12) one gets

$$P_{out}(x) = \|\psi\|^2, \quad |\psi\rangle = \sum_{a=1}^{\chi} z_a |\phi_a\rangle, \quad \phi_a \in \mathcal{S}_t. \tag{13}$$

Here ϕ_a are t-qubit stabilizer states. Below we describe a randomized algorithm that takes as input a t-qubit state ψ, a target error parameter $\epsilon > 0$ and a failure probability p_f. The algorithm computes a real number ξ which, with probability at least $1 - p_f$, approximates the norm of ψ with relative error ϵ. The running time of the algorithm is $O(\chi t^3 \epsilon^{-2} \log(p_f^{-1}))$. The key idea is to approximate $\|\psi\|^2$ by computing inner products between ψ and randomly chosen stabilizer states.

We shall first consider the special case where the failure probability is $1/4$; at the end we describe how to reduce it to a given value p_f^{-1}. Let $\theta \in \mathcal{S}_t$ be a random stabilizer state drawn from the uniform distribution. Define expectation values

$$M_2 \equiv \mathbb{E}_\theta |\langle \theta | \psi \rangle|^2 \quad \text{and} \quad M_4 \equiv \mathbb{E}_\theta |\langle \theta | \psi \rangle|^4.$$

Using Eq. (8) one can compute M_2 and M_4 by pretending that θ is drawn from the Haar measure. Standard formulas for the integrals over the unit sphere yield

$$M_2 = \frac{\|\psi\|^2}{d} \quad \text{and} \quad M_4 = \frac{2\|\psi\|^4}{d(d+1)}, \quad \text{where} \quad d \equiv 2^t. \tag{14}$$

Suppose $\theta_1, \ldots, \theta_L \in \mathcal{S}_t$ are random independent stabilizer states. Define a random variable

$$\xi = \frac{d}{L} \sum_{i=1}^{L} |\langle \theta_i | \psi \rangle|^2. \tag{15}$$

From Eq. (14) one infers that the expected value of ξ is $\bar{\xi} = \mathbb{E}(\xi) = \|\psi\|^2$ and the standard deviation of ξ is

$$\sigma = \sqrt{d^2 L^{-1} (M_4 - M_2^2)} = \sqrt{\frac{d-1}{d+1}} L^{-1/2} \|\phi\|^2.$$

For large t one has $\sigma \approx L^{-1/2} \|\psi\|^2$. By the Chebyshev inequality, $\Pr\big[\, |\xi - \bar{\xi}| \geq 2\sigma \,\big] \leq \frac{1}{4}$. Thus

$$(1-\epsilon)\|\psi\|^2 \leq \xi \leq (1+\epsilon)\|\psi\|^2 \tag{16}$$

with probability at least $3/4$ provided that $L = 4\epsilon^{-2}$.

Now let us discuss how to reduce the failure probability (from $1/4$) so that it is below a given value p_f^{-1}. To achieve this, we compute independent estimates $\xi_1, \xi_2, \ldots \xi_J$ using the above procedure and output the median ξ_{med} of these values. It is a simple fact that this procedure reduces the failure probability to below p_f^{-1} using only $J = O(\log(p_f^{-1}))$ estimates (see Lemma 6.1 of Ref. [20]). With this choice, the probability that Eq. (16) holds with ξ replaced by ξ_{med} is at least $1 - p_f$.

The inner product $\langle \theta_i | \psi \rangle = \sum_{a=1}^{\chi} z_a \langle \theta_i | \phi_a \rangle$ in Eq. (15) can be computed in time $O(\chi t^3)$ since θ_i and ϕ_a are stabilizer states of t qubits. It follows that $P_{out}(x) = \|\psi\|^2$ can be approximated in time $O(\chi t^3 \epsilon^{-2} \log(p_f^{-1}))$, as promised.

Since the runtime grows linearly with χ, we would like to choose a stabilizer decomposition in Eq. (12) with a small rank χ. Clearly, the optimal choice is $\chi = \chi_t$, where

$\chi_t \equiv \chi_t(0)$ is the stabilizer rank defined in the introduction. Unfortunately, the exact value of χ_t is unknown. Using the identity

$$|A^{\otimes 2}\rangle = \frac{1}{2}(|00\rangle + i|11\rangle) + \frac{e^{i\pi/4}}{2}(|01\rangle + |10\rangle) \qquad (17)$$

one can see that $A^{\otimes 2}$ is a linear combination of two stabilizer states, that is, $\chi_2 = 2$. By dividing t qubits into $t/2$ pairs and applying the decomposition Eq. (17) to each pair one gets $\chi_t \le 2^{t/2}$. The results of [15] give a slightly better bound $\chi_t \le 2^{\beta t}$ with $\beta \approx 0.47$. This completes the analysis of the first algorithm.

Remark 1: If $\mathcal{G}$ has a small dimension, namely, $r < \beta$, it can be easier to compute $P_{out}(x)$ directly from Eqs. (7,9) which yield $P_{out}(x) \sim \sum_{P\in\mathcal{G}} \langle A^{\otimes t}|P|A^{\otimes t}\rangle$. Clearly, each term in the sum can be computed in time $O(t)$, so the overall runtime becomes $O(t|\mathcal{G}|) = O(t2^r)$.

Remark 2: An alternative strategy to estimate the expectation value in Eq. (9) is to compute the inner products

$$\langle \Pi_{\mathcal{G}}\theta_i|A^{\otimes t}\rangle = \sum_{a=1}^{\chi} y_a \langle \Pi_{\mathcal{G}}\theta_i|\varphi_a\rangle$$

for $i = 1, \ldots, L$. Here φ_a are the stabilizer states defined in Eq. (12) and θ_i are random stabilizer states. The same arguments as above show that

$$\langle A^{\otimes t}|\Pi_{\mathcal{G}}|A^{\otimes t}\rangle = \|\Pi_{\mathcal{G}}A^{\otimes t}\|^2 \approx \frac{d}{L}\sum_{i=1}^{L} |\langle \Pi_{\mathcal{G}}\theta_i|A^{\otimes t}\rangle|^2.$$

This may be beneficial in the regime $L \ll \chi$ since one has to compute the action of $\Pi_{\mathcal{G}}$ only L times rather than χ times.

Let us now describe the algorithm that allows one to sample x from the distribution P_{out} with statistical error ϵ. As before, we replace each T-gate in the original circuit U by the gadget shown on Fig. 2, prepare all magic states $|A\rangle$ at the very first time step, and permute the qubits such that the initial state is $|0^n A^{\otimes t}\rangle$. Let $y_j \in \{0,1\}$ be the outcome of the measurement performed in the j-th gadget and $y = (y_1, \ldots, y_t)$. Let V_y be the Clifford circuit on $n+t$ qubits corresponding to measurement outcomes y. Each gadget with $y_j = 0$ contributes a CNOT gate to V_y, whereas each gadget with $y_j = 1$ contributes a CNOT and the S-gate to V_y. Thus V_y contains $c+t+|y|$ gates. A composition of all gadgets and Clifford gates of U implements a trace preserving completely positive (TPCP) map

$$\Phi(\rho) = \sum_y (I_n \otimes |y\rangle\langle y|)V_y \rho V_y^\dagger (I_n \otimes |y\rangle\langle y|).$$

Here I_n is the n-qubit identity operator and the sum runs over all t-bit strings y. Suppose first that Φ is applied to a state $\rho_{in} = |0^n\rangle\langle 0^n| \otimes |A\rangle\langle A|^{\otimes t}$. Then the final state of the n computational qubits is $U|0^n\rangle$ regardless of y and each y appears with probability 2^{-t}. Thus

$$\Phi(\rho_{in}) = U|0^n\rangle\langle 0^n|U^\dagger \otimes \frac{I}{2^t}. \qquad (18)$$

Next suppose that Φ is applied to a state $\tilde\rho_{in} = |0^n\rangle\langle 0^n| \otimes |\psi\rangle\langle\psi|$, where ψ is a linear combination of χ stabilizer states $\varphi_1, \ldots, \varphi_\chi \in \mathcal{S}_t$ that approximates $A^{\otimes t}$ with a small error:

$$|\psi\rangle = \sum_{a=1}^{\chi} z_a|\varphi_a\rangle, \qquad |\langle A^{\otimes t}|\psi\rangle|^2 \ge 1 - \epsilon^2/25. \qquad (19)$$

Here z_a are some coefficients and we assume ψ has unit norm. The error $\epsilon^2/25$ is sufficient to ensure that the output distribution of the overall simulation algorithm is ϵ-close to P_{out}. From Eq. (19) one gets

$$\|\rho_{in} - \tilde\rho_{in}\|_1 = \| \, |A\rangle\langle A|^{\otimes t} - |\psi\rangle\langle\psi| \, \|_1 \le \frac{2}{5}\epsilon. \qquad (20)$$

By definition of Φ,

$$\Phi(\tilde\rho_{in}) = \sum_y p_y \, |\phi_y\rangle\langle\phi_y| \otimes |y\rangle\langle y|, \qquad (21)$$

where

$$p_y = \langle 0^n \otimes \psi|V_y^\dagger (I_n \otimes |y\rangle\langle y|)V_y|0^n \otimes \psi\rangle \qquad (22)$$

and ϕ_y are normalized t-qubit states defined by

$$|\phi_y\rangle = p_y^{-1/2}\langle y|V_y|0^n \otimes \psi\rangle. \qquad (23)$$

Clearly, p is a normalized probability distribution on the set of t-bit strings. The state ϕ_y is defined only for $p_y > 0$. Combining Eqs. (18,20,21) and tracing out the last t qubits of $\Phi(\rho_{in})$ and $\Phi(\tilde\rho_{in})$ one infers that

$$\| \, U|0^n\rangle\langle 0^n|U^\dagger - \sum_y p_y|\phi_y\rangle\langle\phi_y| \, \|_1 \le \frac{2}{5}\epsilon. \qquad (24)$$

Here we noted that TPCP maps do not increase the trace distance. Combining Eqs. (18,20,21) and tracing out the first n qubits of $\Phi(\rho_{in})$ and $\Phi(\tilde\rho_{in})$ shows that the distribution p satisfies $\|p - u\|_1 \le \frac{2}{5}\epsilon$, where u is the uniform distribution on the set of t-bit strings. Using this fact and Eq. (24) we arrive at

$$\| \, U|0^n\rangle\langle 0^n|U^\dagger - \frac{1}{2^t}\sum_y |\phi_y\rangle\langle\phi_y| \, \|_1 \le \frac{4}{5}\epsilon. \qquad (25)$$

For each t-bit string y define a probability distribution $P^y_{out}(x) = \langle\phi_y|\Pi(x)|\phi_y\rangle$. Below we give an algorithm which takes as input y and ϵ and produces a sample from a distribution $\tilde P^y_{out}$ which satisfies

$$\|P^y_{out}(x) - \tilde P^y_{out}(x)\|_1 \le \epsilon/5 \qquad (26)$$

Our algorithm to approximately sample from P_{out} has two steps. We first generate a random uniformly distributed t-bit string y and then we sample x from $\tilde P^y_{out}$. From Eqs. (25,26) we see that the distribution over outputs $x \in \{0,1\}^w$ produced by this algorithm approximates P_{out} within error ϵ in the trace norm.

We are now ready to describe how to sample from $\tilde{P}^y_{out}$ satisfying Eq. (26). We first describe how to compute an approximation to $P^y_{out}(x)$ with relative error δ. Note that

$$P^y_{out}(x) = \frac{\langle 0^n \otimes \psi | V_y^\dagger (\Pi(x) \otimes |y\rangle\langle y|) V_y | 0^n \otimes \psi\rangle}{\langle 0^n \otimes \psi | V_y^\dagger (I_n \otimes |y\rangle\langle y|) V_y | 0^n \otimes \psi\rangle}. \quad (27)$$

Here we used Eqs. (22,23). Repeating the same arguments as in the derivation of Eq. (9) one gets

$$P^y_{out}(x) = \frac{2^{-u}\langle\psi|\Pi_{\mathcal{G}}|\psi\rangle}{2^{-v}\langle\psi|\Pi_{\mathcal{H}}|\psi\rangle} \quad (28)$$

for some stabilizer groups $\mathcal{G}, \mathcal{H} \subseteq \mathcal{P}_t$ and integers u, v that can be computed in time $\tau_1 = O((w+t)(c+t) + (n+t)^3)$. We already know a randomized algorithm which computes $\langle\psi|\Pi_{\mathcal{G}}|\psi\rangle$ and $\langle\psi|\Pi_{\mathcal{H}}|\psi\rangle$ with a relative error δ in time $\tau_2 = O(\chi t^3 \delta^{-2} \log(p_f^{-1}))$. Recall that p_f is the probability that the algorithm does not achieve the desired approximation. Thus we can compute $P^y_{out}(x)$ with a relative error 2δ in time $\tau_1 + \tau_2$.

Now consider the task of sampling from P^y_{out}. Assume for simplicity that $Q_{out} = \{1, 2, \ldots, w\}$. For each $j = 1, \ldots, w-1$ define conditional probabilities

$$P^y_{out}(z|x_1, \ldots, x_{j-1}) = \frac{P^y_{out}(x_1, \ldots, x_{j-1}, z)}{P^y_{out}(x_1, \ldots, x_{j-1})}, \quad (29)$$

where $z \in \{0, 1\}$. Suppose the bits $x_1, \ldots, x_{j-1}$ have already been sampled (initially $j = 1$). Then the next bit x_j can be sampled by tossing a coin with bias $P^y_{out}(0|x_1, \ldots, x_{j-1})$. Things are complicated by the fact that we cannot exactly compute this conditional probability. We use the same simulation strategy except that at each step the conditional probability $P^y_{out}(0|x_1, \ldots, x_{j-1})$ is replaced by an approximation q_j. Here we require that with probability at least $1 - p_f$, both q_j and $1 - q_j$ approximate the conditional probabilities $P^y_{out}(0|x_1, \ldots, x_{j-1})$ and $P^y_{out}(1|x_1, \ldots, x_{j-1})$ respectively with relative error $O(\delta)$. Such an approximation q_j can be computed in time $O(\tau_1 + \tau_2)$ using the procedure described above for approximating the probabilities on the right-hand side of Eq. (29). Indeed, first compute a, b which, with probability at least $1 - p_f$, approximate $P^y_{out}(0|x_1, \ldots, x_{j-1})$ and $P^y_{out}(1|x_1, \ldots, x_{j-1})$ respectively with relative error δ. If $a \leq b$ then we set $q_j = a$ while if $b < a$ then we set $q_j = 1 - b$.

We now analyze the resulting simulation algorithm and show that we can ensure Eq. (26) by choosing approximation error $\delta = O(\epsilon w^{-1})$ and failure probability $p_f = O(\epsilon w^{-1})$. Let us first suppose that all probabilities q_j computed by the algorithm achieve the desired approximation δ (i.e., no failures occur). Conditioned on this event we see that the output distribution produced by the algorithm approximates $P^y_{out}(x)$ with relative error $O(\delta w)$. This conditional probability distribution can therefore be made $\epsilon/10$-close (say) to P^y_{out} by choosing $\delta = O(\epsilon w^{-1})$. It remains to show that by choosing $p_f = O(\epsilon w^{-1})$ we can ensure that the output distribution $\tilde{P}^y_{out}$ of the simulation algorithm is $\epsilon/10$-close to the distribution conditioned on no failures. This follows because the algorithm computes $O(w)$ probabilities $\{q_j\}$ in total and choosing $p_f = O(\epsilon w^{-1})$ we can ensure that all of them are computed to within the desired approximation error δ, with probability at least $1 - \epsilon/20$. With this choice we have $\tilde{P}^y_{out} = (1 - \epsilon/20)P_A + \epsilon/20 P_B$ where P_A is the distribution conditioned on no failures, and thus $\|\tilde{P}^y_{out} - P_A\|_1 \leq \epsilon/10$ as claimed.

The overall running time of this algorithm is $\tau_1' + \tau_2'$, where $\tau_1' = O(w\tau_1) = O(w(w+t)(c+t) + w(n+t)^3)$ and $\tau_2' = O(w\tau_2) = O(\chi w^3 t^3 \epsilon^{-2} \log(w\epsilon^{-1}))$.

Remark: This algorithm can be modified slightly to handle certain Clifford+T circuits which use measurement and classical control. To see how, recall that in the T-gate gadget from Fig. 2, a single qubit is measured in the computational basis (yielding both outcomes with equal probability) and a Clifford operation is classically controlled on the measurement outcome. In our simulation algorithm the measurement is replaced by a uniformly chosen postselection bit y_j. Exactly the same strategy can be used for other simple gadgets which involve measurement and classical control. For example, the Toffoli gate can be implemented as a Clifford+T circuit with only four T-gates if we allow two ancillas, measurement, and classical control [13](otherwise it requires seven T-gates [21, 22]). Fortunately it is possible to use the less costly circuit with four T-gates in the above simulation algorithm by including one additional postselection bit per Toffoli gate.

V. APPROXIMATING MAGIC STATES

In this section we show how to compute a decomposition Eq. (19) with $\chi = O(2^{\gamma t}\epsilon^{-2})$, where γ satisfies Eq. (4). Define a state

$$|H\rangle = \cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle.$$

We note that the magic state $|A\rangle$ is equivalent to $|H\rangle$ modulo Clifford gates and a global phase, $|A\rangle = e^{i\pi/8} H S^\dagger |H\rangle$. Thus it suffices to construct a state

$$|\psi\rangle = \sum_{a=1}^{\chi} z_a |\varphi_a\rangle, \quad \varphi_1, \ldots, \varphi_\chi \in \mathcal{S}_t \quad (30)$$

such that $\|\psi\| = 1$,

$$|\langle H^{\otimes t}|\psi\rangle|^2 \geq 1 - \delta \quad \text{and} \quad \chi = O(2^{\gamma t}\delta^{-1}) \quad (31)$$

for all sufficiently small $\delta > 0$.

Our starting point is the identity

$$|H^{\otimes t}\rangle = \frac{1}{(2\nu)^t} \sum_{x \in \mathbb{F}_2^t} |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \ldots \otimes \tilde{x}_t\rangle \quad (32)$$

where $|\tilde{0}\rangle \equiv |0\rangle$, $|\tilde{1}\rangle \equiv H|0\rangle = 2^{-1/2}(|0\rangle + |1\rangle)$, and

$$\nu \equiv \cos(\pi/8).$$

The right-hand side of Eq. (32) is a uniform superposition of 2^t non-orthogonal stabilizer states labeled by elements of the vector space $\mathbb{F}_2^t$. We shall construct an approximation ψ which is a uniform superposition of states $|\tilde{x}_1 \otimes \tilde{x}_2 \otimes \ldots \otimes \tilde{x}_t\rangle$ over a linear subspace of $\mathbb{F}_2^t$.

Let $L(t,k)$ be the set of all k-dimensional linear subspaces $\mathcal{L} \subseteq \mathbb{F}_2^t$. We will fix k below. For each $\mathcal{L} \in L(t,k)$ define a state

$$|\mathcal{L}\rangle = \frac{1}{\sqrt{2^k Z(\mathcal{L})}} \sum_{x \in \mathcal{L}} |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \cdots \otimes \tilde{x}_t\rangle \qquad (33)$$

where

$$Z(\mathcal{L}) \equiv \sum_{x \in \mathcal{L}} 2^{-|x|/2}. \qquad (34)$$

Using the identity $\langle \tilde{a}|\tilde{b}\rangle = 2^{-|a \oplus b|/2}$, where $a, b \in \{0,1\}$, and the fact that $\mathcal{L}$ is a linear subspace one can easily check that $|\mathcal{L}\rangle$ is a normalized state, $\langle \mathcal{L}|\mathcal{L}\rangle = 1$. We take our approximation ψ from Eq. (30) to be Eq. (33) for a suitably chosen subspace $\mathcal{L}^\star \in L(t,k)$, which gives an approximate decomposition of $H^{\otimes t}$ using $\chi = 2^k$ stabilizer states. How small can we hope to make k? Using the fact that $\langle H|\tilde{0}\rangle = \langle H|\tilde{1}\rangle = \nu$ we see that

$$|\langle H^{\otimes t}|\mathcal{L}\rangle|^2 = \frac{2^k \nu^{2t}}{Z(\mathcal{L})} \qquad (35)$$

From this we immediately get a lower bound on k. Indeed, since $Z(\mathcal{L}) \geq 1$ we will need $2^k \geq \nu^{-2t}(1-\delta)$ to achieve the desired approximation. Below we describe a randomized algorithm which outputs a subspace $\mathcal{L}^\star$ with $2^k = O(\delta^{-1}\nu^{-2t})$. Thus for constant δ we achieve the best possible scaling of k with t. We will use the following fact about random subspaces of $\mathbb{F}_2^t$.

Lemma 1. *Let $\mathcal{L} \in L(t,k)$ be chosen uniformly at random. Then*

$$\mathbb{E}(Z(\mathcal{L})) \leq 1 + 2^k \nu^{2t}. \qquad (36)$$

Proof. By linearity, we have

$$\mathbb{E}(Z(\mathcal{L})) = 1 + \sum_{x \in \mathbb{F}_2^t \setminus 0} 2^{-|x|/2} \cdot \mathbb{E}(\chi_{\mathcal{L}}(x)), \qquad (37)$$

where $\chi_{\mathcal{L}}(x)$ is the indicator function of $\mathcal{L}$. The expectation value $\mathbb{E}(\chi_{\mathcal{L}}(x))$ with respect to $\mathcal{L}$ for a fixed x is $(2^k-1)/(2^t-1)$. Thus we arrive at

$$\begin{aligned}\mathbb{E}(Z(\mathcal{L})) &= 1 + \frac{(2^k-1)}{(2^t-1)} \sum_{x \in \mathbb{F}_2^t \setminus 0} 2^{-|x|/2} \\ &= 1 + \frac{(2^k-1)}{(2^t-1)} \left(2^t \nu^{2t} - 1\right) \\ &\leq 1 + 2^k \nu^{2t}.\end{aligned}$$

□

As a corollary, there exists at least one $\mathcal{L} \in L(t,k)$ such that $Z(\mathcal{L}) \leq 1 + 2^k \nu^{2t}$. We now fix k to be the unique positive integer satisfying

$$4 \geq 2^k \nu^{2t} \delta \geq 2. \qquad (38)$$

Consider a subspace $\mathcal{L} \in L(t,k)$ chosen uniformly at random. Using Markov's inequality and Lemma 1 we get

$$\begin{aligned}\Pr\left[\frac{Z(\mathcal{L})}{(1+2^k\nu^{2t})(1+\delta/2)} \geq 1\right] &\leq \frac{\mathbb{E}(Z(\mathcal{L}))}{(1+2^k\nu^{2t})\,(1+\delta/2)} \\ &\leq 1 - \frac{\delta}{2+\delta}.\end{aligned}$$

For a given $\mathcal{L} \in L(t,k)$ we may compute $Z(\mathcal{L})$ in time $O(2^k)$. By randomly choosing $O(1/\delta)$ subspaces $\mathcal{L}$ we obtain one $\mathcal{L}^\star$ satisfying

$$Z(\mathcal{L}^\star) \leq (1+2^k\nu^{2t})(1+\delta/2) \qquad (39)$$

with constant probability. Plugging Eq. (39) into Eq. (35) we see that

$$\begin{aligned}|\langle H^{\otimes t}|\mathcal{L}^\star\rangle|^2 &\geq \frac{1}{(1+2^{-k}\nu^{-2t})\,(1+\delta/2)} \\ &\geq \frac{1}{(1+\delta/2)^2} \\ &\geq 1-\delta,\end{aligned}$$

where in the second line we used Eq. (38). The state $|\psi\rangle = |\mathcal{L}^\star\rangle$ obtained in this way therefore satisfies Eq. (31) with

$$\chi = 2^k \leq 4\nu^{-2t}\delta^{-1} = O(\nu^{-2t}\delta^{-1}) = O(2^{\gamma t}\delta^{-1}). \qquad (40)$$

This algorithm has running time $O(\nu^{-2t}\delta^{-2})$, since we must check the condition Eq. (39) for each of the $O(\delta^{-1})$ randomly sampled elements of $L(t,k)$ (note that the time required to sample each element is $O(poly(t))$).

Remark: One may ask whether a stronger bound on χ can be obtained by truncating the expansion of $H^{\otimes t}$ in some other basis of stabilizer states. For example, consider the standard $0,1$-basis of t qubits. The expansion of $H^{\otimes t}$ in this basis is concentrated on basis vectors $x \in \mathbb{F}_2^t$ with Hamming weight $|x| = (1-\nu^2)t \pm O(t^{1/2})$. The number of such basis vectors scales as $\chi \sim 2^{tH_2(\nu^2)} \approx 2^{0.6t}$, where $H_2(p)$ is the binary Shannon entropy function. Thus replacing the $\tilde{0}, \tilde{1}$-basis by the $0,1$-basis gives a significantly worse bound on χ.

As noted above, taking δ to be a constant our construction has the best possible scaling $\chi = O(\nu^{-2t})$ of any decomposition of the form Eq. (33). In fact, we prove the following lower bound on the stabilizer rank of $H^{\otimes t}$.

Lemma 2. *Consider a state $|\psi\rangle = \sum_{a=1}^{\chi} z_a |\phi_a\rangle$, where $\phi_a \in \mathcal{S}_t$. Suppose $\|\psi\| = 1$ and $|\langle \psi|H^{\otimes t}\rangle| \geq f$. Then $\chi \geq \nu^{-2t} f^2 \|z\|^{-2}$, where $z = (z_1, \ldots, z_\chi) \in \mathbb{C}^\chi$.*

Proof. First, let us show that

$$F_t \equiv \max_{\phi \in \mathcal{S}_t} |\langle \phi|H^{\otimes t}\rangle| = \nu^t. \qquad (41)$$

The lower bound $F_t \geq \nu^t$ is obvious since $\langle 0^{\otimes t}|H^{\otimes t}\rangle = \nu^t$. We shall use induction in t to show that $F_t \leq \nu F_{t-1}$. Consider some fixed t and let $F_t = |\langle\phi|H^{\otimes t}\rangle|$ for some $\phi \in \mathcal{S}_t$. Suppose we measure the first qubit of ϕ in the $0,1$ basis. Let P_a be the probability of getting the outcome $a = 0, 1$. It is well-known that $P_a \in \{0, 1, 1/2\}$ for any stabilizer state ϕ. Consider three cases.
Case 1: $P_0 = 1$. Then $|\phi\rangle = |0\rangle \otimes |\psi\rangle$ for some $\psi \in \mathcal{S}_{t-1}$ and $F_t = \nu|\langle\psi|H^{\otimes(t-1)}\rangle| \leq \nu F_{t-1}$.
Case 2: $P_0 = 0$. Then $|\phi\rangle = |1\rangle \otimes |\psi\rangle$ for some $\psi \in \mathcal{S}_{t-1}$ and $F_t = \sqrt{1-\nu^2}|\langle\psi|H^{\otimes(t-1)}\rangle| < \nu F_{t-1}$.
Case 3: $P_0 = 1/2$. Then

$$|\phi\rangle = 2^{-1/2}\left(|0\rangle \otimes |\psi_0\rangle + |1\rangle \otimes |\psi_1\rangle\right)$$

for some $\psi_0, \psi_1 \in \mathcal{S}_{t-1}$. By triangle inequality,

$$F_t \leq 2^{-1/2}(\nu + \sqrt{1-\nu^2})F_{t-1} = \nu F_{t-1}.$$

The base of induction $F_1 = \nu$ is trivial. This proves Eq. (41). From Eq. (41) one gets

$$f \leq |\langle\psi|H^{\otimes t}\rangle| \leq \nu^t \sum_{a=1}^{\chi} |z_a| \leq \nu^t \chi^{1/2}\|z\|.$$

This is equivalent to the statement of the lemma. □

We conjecture that any approximate stabilizer decomposition of $H^{\otimes t}$ that achieves a constant approximation error must use at least $\Omega(\nu^{-2t})$ stabilizer states.

VI. ACKNOWLEDGMENTS

DG acknowledges funding provided by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (NFS Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028). SB thanks Alexei Kitaev for helpful discussions and comments.

APPENDIX A: QUADRATIC FORMS

The remaining sections provide more details on implementation of our algorithms. Appendix A presents some basic facts about quadratic forms over finite fields and describes a subroutine for computing certain exponential sums. The standard form of stabilizer states used in all our algorithms is defined in Appendix B. Then we present algorithms for computing the inner product between stabilizer states (Appendix C), generating a random uniformly distributed stabilizer state (Appendix D), and computing the action of Pauli measurements on stabilizer states (Appendix E). The three algorithms have running time $O(n^3)$, $O(n^2)$, and $O(n^2)$ respectively, where n is the number of qubits. We provide pseudocode for all algorithms and report timing analysis for a MATLAB implementation. Appendix F describes simulation of the hidden shift algorithm.

Below we consider functions that map binary vectors to integers modulo eight. We define a special class of such functions that are analogous to quadratic forms over the real field. The definition of $\mathbb{Z}_8$-valued quadratic forms given below was proposed to us by Kitaev [23]. Analogous definitions and computations using $\mathbb{Z}_4$-valued quadratic forms can be found in [24]. For a general theory of quadratic forms over a finite field see Ref. [25]. Throughout the rest of the paper arithmetic operations $\pm$ are performed modulo eight (unless stated otherwise), whereas addition of binary vectors modulo two is denoted $\oplus$. Elements of $\mathbb{F}_2^n$ are considered as binary row vectors. A binary inner product between vectors $x, y \in \mathbb{F}_2^n$ will be denoted $(x, y) \equiv \sum_{i=1}^n x_i y_i \pmod 2$. A set of binary matrices of size $a \times b$ is denoted $\mathbb{F}_2^{a\times b}$. A transpose of a matrix M is denoted M^T.

Recall that a subset $\mathcal{K} \subseteq \mathbb{F}_2^n$ is a called an affine space of dimension k iff $\mathcal{K} = \mathcal{L}(\mathcal{K}) \oplus h$ for some k-dimensional linear subspace $\mathcal{L}(\mathcal{K}) \subseteq \mathbb{F}_2^n$ and a shift vector $h \in \mathbb{F}_2^n$. Note that $\mathcal{K}$ uniquely determines $\mathcal{L}(\mathcal{K})$, namely, $\mathcal{L}(\mathcal{K}) = \{x \oplus y : x, y \in \mathcal{K}\}$. The shift vector h however is not uniquely defined. Obviously, $|\mathcal{K}| = |\mathcal{L}(\mathcal{K})| = 2^k$.

Definition 1. *Consider an affine space $\mathcal{K} \subseteq \mathbb{F}_2^n$. A function $q : \mathcal{K} \to \mathbb{Z}_8$ is called a quadratic form iff there exists a function $J : \mathcal{L}(\mathcal{K}) \times \mathcal{L}(\mathcal{K}) \to \mathbb{Z}_8$ such that*

$$q(x \oplus y \oplus z) + q(z) - q(x \oplus z) - q(y \oplus z) = J(x, y) \quad (42)$$

for all $z \in \mathcal{K}$ and for all $x, y \in \mathcal{L}(\mathcal{K})$.

Informally, Eq. (42) demands that a discrete analogue of the second derivative $\frac{\partial^2 q}{\partial x \partial y}$ evaluated at some point $z \in \mathcal{K}$ does not depend on z, as it is the case for quadratic forms over the real field. The next lemma states properties of the function $J(x, y)$ that follow from Eq. (42).

Lemma 3. *The function $J(x, y)$ defined by Eq. (42) is a symmetric bilinear form that takes values $0, 4 \pmod 8$. Namely, $J(x, y) = J(y, x)$, $J(0, y) = 0$, $J(x' \oplus x'', y) = J(x', y) + J(x'', y)$, and $J(x, y) = 0, 4 \pmod 8$ for all $x, x', x'', y \in \mathcal{L}(\mathcal{K})$.*

Proof. Let $x = x' \oplus x''$. Substituting $x \leftarrow x'$ in Eq. (42) gives

$$J(x', y) = q(x' \oplus y \oplus z) + q(z) - q(x' \oplus z) - q(y \oplus z).$$

Substituting $x \leftarrow x''$ and $z \leftarrow z \oplus x'$ in Eq. (42) gives

$$J(x'', y) = q(x \oplus y \oplus z) + q(x' \oplus z) - q(x \oplus z) - q(x' \oplus y \oplus z).$$

This shows that $J(x, y) = J(x', y) + J(x'', y)$. The identities $J(0, y) = 0$ and $J(x, y) = J(y, x)$ follow trivially from Eq. (42). Replacing z by $z \oplus x$ in Eq. (42) yields

$$J(x, y) = q(y \oplus z) + q(x \oplus z) - q(z) - q(x \oplus y \oplus z).$$

Combining this and Eq. (42) one gets $2J(x, y) = 0$, that is, $J(x, y) = 0 \pmod 4$. □

As a corollary, one gets $q(x \oplus z) - q(z) \in \{0, 2, 4, 6\}$ for all $z \in \mathcal{K}$ and for all $x \in \mathcal{L}(\mathcal{K})$. This can be checked by choosing $x = y$ in Eq. (42) and using the fact that $J(x, x) \in \{0, 4\}$.

Suppose $g^1, \ldots, g^k \in \mathcal{L}(\mathcal{K})$ is some fixed basis of $\mathcal{L}(\mathcal{K})$, $h \in \mathcal{K}$ is some fixed shift vector, and $x \in \mathcal{K}$. Then

$$x = h \oplus x_1 g^1 \oplus \ldots \oplus x_k g^k, \quad x_i \in \{0, 1\}.$$

We shall write $\vec{x} \equiv (x_1, \ldots, x_k)$ to avoid confusion between a point $x \in \mathcal{K}$ and its coordinates. Applying Eq. (42) and Lemma 3 one can describe q in a basis-dependent way as

$$q(\vec{x}) = Q + \sum_{a=1}^{k} D_a x_a + \sum_{1 \le a < b \le k} J_{a,b} x_a x_b, \tag{43}$$

where $Q \equiv q(h) \in \mathbb{Z}_8$,

$$D_a = q(g^a \oplus h) - q(h) \in \{0, 2, 4, 6\}, \tag{44}$$

$$J_{a,b} = J_{b,a} = J(g^a, g^b) \in \{0, 4\}. \tag{45}$$

We shall consider J as a symmetric $k \times k$ matrix. Although Eq. (43) depends only on off-diagonal matrix elements of J, it will be convenient to retain the diagonal of J. Combining Eqs. (42,43) one gets

$$J_{a,a} = 2D_a, \quad 1 \le a \le k. \tag{46}$$

A connection between quadratic forms and stabilizer states is established by the following lemma.

Lemma 4. *Any n-qubit stabilizer state can be uniquely written as*

$$|\mathcal{K}, q\rangle \equiv 2^{-k/2} \sum_{x \in \mathcal{K}} e^{\frac{i\pi}{4} q(x)} |x\rangle, \tag{47}$$

where $\mathcal{K} \subseteq \mathbb{F}_2^n$ is an affine space of dimension $0 \le k \le n$ and $q : \mathcal{K} \to \mathbb{Z}_8$ is a quadratic form.

Proof. The claim that any stabilizer state can be written in the form Eq. (47) follows from the explicit characterization of quadratic forms Eqs. (43,44,45) and the canonical form of stabilizer states derived in Refs. [5, 26, 27]. The uniqueness of the decomposition Eq. (47) is obvious. □

Next let us describe how the representation of q transforms under various basis changes. Suppose $R \in \mathbb{F}_2^{k \times k}$ is an invertible matrix. Consider a basis change

$$g^a \leftarrow \sum_{b=1}^{k} R_{a,b}\, g^b \pmod 2, \tag{48}$$

where $1 \le a \le k$. The shift vector h remains unchanged. Applying Eq. (43) where $\vec{x}$ is chosen as the a-th row of R, one can easily check that the coefficients (Q, D, J) transform according to $Q \leftarrow Q$,

$$D_a \leftarrow \sum_{b=1}^{k} R_{a,b} D_b + \sum_{1 \le b < c \le k} J_{b,c} R_{a,b} R_{a,c}, \tag{49}$$

and

$$J \leftarrow R J R^T. \tag{50}$$

The matrix multiplications are performed in the ring $\mathbb{Z}_8$. Next consider a basis change that alters the shift vector,

$$h \leftarrow h \oplus y, \quad \text{where} \quad y = \sum_{a=1}^{k} y_a g^a \pmod 2. \tag{51}$$

Using Eq. (44) one can easily check that the coefficients (Q, D, J) transform according to

$$Q \leftarrow Q + \sum_{a=1}^{k} D_a y_a + \sum_{1 \le a < b \le k} J_{a,b} y_a y_b, \tag{52}$$

$$D_a \leftarrow D_a + \sum_{b=1}^{k} J_{a,b} y_b, \tag{53}$$

and $J \leftarrow J$.

The above rules determine the representation (Q, D, J) of q in any basis of $\mathcal{K}$. What is the cost of computing this representation ? Clearly, all updates can be expressed as a constant number of matrix-matrix (matrix-vector) multiplications with $\mathbb{Z}_8$-valued matrices of size k. Thus the updates have cost $O(k^3)$ in the worst case. We shall often consider basis changes Eq. (48) such that the matrix R is sparse. Let $|R|$ be the total number of non-zeros in R. Using sparse matrix-matrix multiplication one can perform all updates in Eqs. (49,50) in time $O(|R|^2)$. Indeed, let w_a be the number of non-zeros in the a-th row of R. One can update D_a and $J_{a,b}$ for any fixed a, b in time w_a^2 and $w_a w_b$ respectively. Thus D and J can be updated in time $O((\sum_{a=1}^{k} w_a)^2) = O(|R|^2)$. Since the updates Eq. (53,52) require time $O(k^2)$ and $|R| \ge k$, the overall time is $O(|R|^2)$. We conclude that computing the representation (Q, D, J) of q in the new basis takes time

$$\tau_{update} = O(\min(k^3, |R|^2)). \tag{54}$$

In the rest of this section we show how to compute certain exponential sums associated with quadratic forms, namely,

$$W(q) \equiv \sum_{x \in \mathbb{F}_2^k} e^{i\frac{\pi}{4} q(\vec{x})}, \tag{55}$$

where $q(\vec{x})$ is defined by Eq. (43). Of course, the addition in Eq. (55) is over the complex field. Our algorithm takes as input the data k, Q, D, J describing $q(\vec{x})$ and outputs

$W(q)$. The algorithm has running time $O(k^3)$. It will be used as a subroutine for computing the inner product between two stabilizer states, see Appendix C.

It will be convenient to consider a more general sum

$$W(\mathcal{K},q)=\sum_{x\in\mathcal{K}} e^{i\frac{\pi}{4}q(x)}, \tag{56}$$

where $\mathcal{K}=\mathcal{L}(\mathcal{K})\oplus h$ is an affine space and $q:\mathcal{K}\to\mathbb{Z}_8$ is a quadratic form on $\mathcal{K}$. Clearly, Eq. (55) is a special case of Eq. (56). Let us say that $g^1,\ldots,g^k\in\mathcal{L}(\mathcal{K})$ is a *canonical basis* of $\mathcal{L}(\mathcal{K})$ iff the set of basis vectors can be partitioned into disjoint subsets

$$[k]=\mathcal{D}_1\cup\ldots\cup\mathcal{D}_r\cup M\cup S, \tag{57}$$

such that

$$|\mathcal{D}_1|=\ldots=|\mathcal{D}_r|=2,\quad |S|\le 1, \tag{58}$$

$$J_{a,a}=\begin{cases} 0 & \text{if } a\notin S,\\ 4 & \text{if } a\in S.\end{cases} \tag{59}$$

$$a\in M \quad\Rightarrow\quad J_{a,b}=0\quad \forall b\in[k]\setminus S, \tag{60}$$

and $\mathcal{D}_i=\{a,b\}$ implies

$$J_{a,b}=4\quad\text{and}\quad J_{a,c}=J_{b,c}=0\quad\forall c\notin S\cup\{a,b\}. \tag{61}$$

Some of the subsets in Eq. (57) can be empty. Let us write

$$\mathcal{D}_j=\{a(j),b(j)\},\quad j=1,\ldots,r.$$

Assume that $\mathcal{L}(\mathcal{K})$ is already equipped with a canonical basis $g^1,\ldots,g^k$ and show how to compute the sum $W(\mathcal{K},q)$. Suppose first $S=\emptyset$. By repeatedly applying Eq. (43) and using Eqs. (58,60,61) one can check that

$$q(\vec{x})=Q+\sum_{j=1}^{r} q_j(x_{a(j)},x_{b(j)})+\sum_{c\in M} D_c x_c. \tag{62}$$

where $q_j:\mathbb{F}_2^2\to\mathbb{Z}_8$ is defined by

$$q_j(y,z)=4yz+D_{a(j)}y+D_{b(j)}z. \tag{63}$$

Examination of Eqs. (56,62,63) reveals that the sum $W(\mathcal{K},q)$ factorizes into a product of $O(k)$ terms such that each term can be computed in time $O(1)$. Specifically,

$$W(\mathcal{K},q)=e^{i\frac{\pi}{4}Q}\cdot\prod_{c\in M}\left(1+e^{i\frac{\pi}{4}D_c}\right)\cdot\prod_{j=1}^{r}\Gamma_j, \tag{64}$$

where

$$\Gamma_j=1+e^{i\frac{\pi}{4}D_{a(j)}}+e^{i\frac{\pi}{4}D_{b(j)}}-e^{i\frac{\pi}{4}(D_{a(j)}+D_{b(j)})} \tag{65}$$

Combining Eqs. (64,65) one can compute $W(\mathcal{K},q)$ in time $O(k)$.

Consider now the remaining case $S\neq\emptyset$. Since $|S|\le 1$, we have $S=\{s\}$ for some $s\in[k]$. By repeatedly applying Eq. (43) and using Eqs. (58,60,61) one can check that

$$q(\vec{x})=Q+D_s x_s+\sum_{j=1}^{r} q_j(x_{a(j)},x_{b(j)},x_s)+\sum_{c\in M}(D_c x_c+J_{c,s}x_c x_s), \tag{66}$$

where $q_j:\mathbb{F}_2^3\to\mathbb{Z}_8$ is defined by

$$q_j(y,z,\sigma)=4yz+J_{a(j),s}y\sigma+J_{b(j),s}z\sigma+D_{a(j)}y+D_{b(j)}z. \tag{67}$$

We have $W(\mathcal{K},q)=W_0+W_1$, where

$$W_\sigma\equiv\sum_{x\in\mathcal{K}:\,x_s=\sigma} e^{i\frac{\pi}{4}q(\vec{x})},\quad \sigma=0,1. \tag{68}$$

Examination of Eqs. (66,67) reveals that W_σ factorizes into a product of $O(k)$ terms such that each term can be computed in time $O(1)$. Specifically,

$$W_\sigma=e^{i\frac{\pi}{4}(Q+\sigma D_s)}\prod_{c\in M}\left(1+e^{i\frac{\pi}{4}(D_c+\sigma J_{c,s})}\right)\cdot\prod_{j=1}^{r}\Gamma_j(\sigma), \tag{69}$$

where

$$\begin{aligned}\Gamma_j(\sigma)=1&+\exp\left[i\frac{\pi}{4}\left(J_{a(j),s}\sigma+D_{a(j)}\right)\right]\\ &+\exp\left[i\frac{\pi}{4}\left(J_{b(j),s}\sigma+D_{b(j)}\right)\right]\\ &-\exp\left[i\frac{\pi}{4}\left(J_{a(j),s}\sigma+J_{b(j),s}\sigma+D_{a(j)}+D_{b(j)}\right)\right].\end{aligned} \tag{70}$$

Combining Eqs. (69,70) one can compute W_0+W_1 in time $O(k)$.

To transform an arbitrary basis $g^1,\ldots,g^k$ of $\mathcal{L}(\mathcal{K})$ into the canonical form we shall use a version of the Gram-Schmidt orthogonalization. It involves at most k basis changes Eq. (48) with sparse matrices R such that $|R|=O(k)$. Computing the coefficients (D,J) in the canonical basis thus takes time $O(k|R|^2)=O(k^3)$, see Eq. (54).

Recall that $D_a\in\{0,2,4,6\}$. Define a subset

$$S=\{a\in[k]\,:\,D_a\in\{2,6\}\}.$$

If S is non-empty, pick an arbitrary element $s\in S$. Perform a basis change $g^a\leftarrow g^a\oplus g^s$ for each $a\in S\backslash s$. From Eq. (49) one gets $D_a\leftarrow D_a+D_s+J_{a,s}\in\{0,4\}$ for all $a\in S\backslash s$ and $D_a\leftarrow D_a$ for all $a\notin S$. Set $S=\{s\}$. Now we can assume that $D_a\in\{0,4\}$ for all $a\notin S$ for some subset S such that $|S|\le 1$. From Eq. (46) we infer

$$J_{a,a}=0\quad\text{for all } a\notin S. \tag{71}$$

Let us say that a pair of basis vectors (g^a,g^b) with $a,b\notin S$ is a dimer if it obeys Eq. (61), that is, $J_{a,b}=4$ and $J_{a,c}=J_{b,c}=0$ for all $c\notin S\cup\{a,b\}$. Note that a basis vector can belong to at most one dimer. Let us say that a basis vector g^a with $a\notin S$ is a monomer if it obeys

Eq. (60), that is, $J_{a,b} = 0$ for all $b \in [k] \setminus S$. Partition the set of basis vectors into four disjoint sets,

$$[k] = \mathcal{D} \cup M \cup S \cup E, \tag{72}$$

such that $\mathcal{D}$ is the union of all dimers, M is the union of all monomers, and E is the complement of $\mathcal{D}MS$. By definition, a basis has a canonical form iff E is empty. Initially $\mathcal{D}$, M are empty, and E is the complement of S. Suppose E is non-empty. Pick any $a \in E$. If $J_{a,b} = 0$ for all $b \in E$, move a from E to M. Otherwise $J_{a,b} = 4$ for some $b \in E$. Let us define a binary matrix $\boldsymbol{J}$ corresponding to J such that $\boldsymbol{J}_{a,b} = 1$ if $J_{a,b} = 4$ and $\boldsymbol{J}_{a,b} = 0$ otherwise. Perform a basis change

$$g^c \leftarrow g^c \oplus \boldsymbol{J}_{a,c} g^b \oplus \boldsymbol{J}_{b,c} g^a \quad \text{for all } c \in E \setminus \{a,b\}. \tag{73}$$

Using Eq. (71) one can check that the new basis vectors obey $J(g^c, g^a) = J(g^c, g^b) = 0$ for all $c \in \mathcal{D}ME \setminus \{a,b\}$. Thus we can move a, b from E to $\mathcal{D}$ by creating a new dimer $\mathcal{D}_i = \{a,b\}$ in Eq. (57). By repeating the above steps at most k times one makes $E = \emptyset$. Furthermore, the R matrices corresponding to the basis change Eq. (73) are sparse since any row of R contains at most three non-zero elements. Thus the original basis is transformed into the canonical form by $O(k)$ basis changes Eq. (48) with sparse matrices R such that $|R| = O(k)$. This has cost $O(k|R|^2) = O(k^3)$. We summarize the algorithm below.

```
function ExponentialSum(Q, D, J)
    S ← {a ∈ [k] : D_a ∈ {2,6}}
    if S ≠ ∅ then
        Pick any a ∈ S
        for b ∈ S \ {a} do
            g^b ← g^b ⊕ g^a
        end for
        Update (D, J) using Eqs. (49,50)
        S ← {a}
    end if
                        ▷ Now J_{a,a} = 0 for all a ∉ S
    E ← [k] \ S
    M ← ∅
    r ← 0
    while E ≠ ∅ do
        Pick any a ∈ E
        K ← {b ∈ E \ a : J_{a,b} = 4}
        if K = ∅ then
                            ▷ Found a new monomer {a}
            M ← M ∪ a
            E ← E \ a
        else
            Pick any b ∈ K
            for c ∈ E \ {a,b} do
                g^c ← g^c ⊕ J_{a,c} g^b ⊕ J_{b,c} g^a
            end for
            Update (D, J) using Eqs. (49,50)
                    ▷ Now {a,b} form a new dimer
            r ← r + 1, D_r ← {a,b}
            E ← E \ {a,b}
        end if
    end while
    if S = ∅ then
        Compute W(K, q) from Eq. (64)
    else
        Compute W_{0,1} from Eq. (69)
        Set W(K, q) = W_0 + W_1
    end if
end function
```

Comments: The basis vectors g^a only serve a notational purpose to describe the basis change matrix R that must be used in the update formulas Eqs. (49,50). There are no actual data representing g^a or operations performed with them. For example, the first **for** loop corresponds to a matrix $R = I \oplus \sum_{b \in S \setminus \{a\}} (e^b)^T e^a$, where e^a is the binary vector with a single '1' at the a-th position. As was shown in Ref. [15], the sum $W(\mathcal{K}, q)$ can be represented by a triple of integers $p \geq 0$, $m \in \mathbb{Z}_8$, and $\epsilon \in \{0,1\}$ such that $W(\mathcal{K}, q) = \epsilon \cdot 2^{p/2} \cdot e^{i\pi m/4}$. Our implementation of the algorithm uses such representation for all intermediate sums to avoid roundoff errors. Timing analysis for a MATLAB implementation is reported in Table 1.

APPENDIX B: STANDARD FORM OF STABILIZER STATES

Suppose $|\mathcal{K}, q\rangle \in \mathcal{S}_n$ is a stabilizer state of n qubits defined in Eq. (47). An affine space $\mathcal{K} = \mathcal{L}(\mathcal{K}) \oplus h \subseteq \mathbb{F}_2^n$ of dimension k will be represented by a tuple

$$(n, k, h \in \mathbb{F}_2^n, G, \bar{G} \in \mathbb{F}_2^{n\times n}),$$

such that $\mathcal{L}(\mathcal{K})$ is spanned by the first k rows of the matrix G and $\bar{G} \equiv (G^{-1})^T$, that is,

$$G\bar{G}^T = I \pmod 2. \tag{74}$$

We shall write g^a and $\bar{g}^a$ for the a-th row of G and $\bar{G}$ respectively. Thus $\mathcal{L}(\mathcal{K}) = \mathrm{span}(g^1, \ldots, g^k)$ and $(g^a, \bar{g}^b) = \delta_{a,b}$ for $1 \le a, b \le n$. We shall refer to g^a and $\bar{g}^a$ as the primal and the dual basis vectors.

A quadratic form $q : \mathcal{K} \to \mathbb{Z}_8$ will be specified by a list of coefficients (Q, D, J) that describe $q(\vec{x})$ in the basis $g^1, \ldots, g^k$ of $\mathcal{L}(\mathcal{K})$, see Eqs. (43,44,45), with the shift vector h. Thus, a stabilizer state $|\mathcal{K}, q\rangle$ of n qubits is described by the following data:

$$(n, k, h, G, \bar{G}, Q, D, J),$$

where $Q \in \mathbb{Z}_8$, $D_1, \ldots, D_k \in \{0, 2, 4, 6\}$, and J is a symmetric $k \times k$ such that $J_{a,b} \in \{0, 4\}$ for all a, b. A valid data must satisfy conditions Eq. (74) and Eq. (46).

We shall often use a subroutine that alters a stabilizer state $|\mathcal{K}, q\rangle$ by shrinking the affine space $\mathcal{K}$ reducing its dimension by one. Namely, consider a vector $\xi \in \mathbb{F}_2^n$ and $\alpha \in \mathbb{F}_2$. Define

$$\mathcal{M} = \mathcal{K} \cap \{x \in \mathbb{F}_2^n : (\xi, x) = \alpha\}. \tag{75}$$

Clearly, $\mathcal{M}$ is an affine space which is either empty, or $\mathcal{M} = \mathcal{K}$, or $\mathcal{M}$ has dimension $k - 1$. Below we describe an algorithm that takes as input a stabilizer state $|\mathcal{K}, q\rangle$ and computes the standard form of the state $|\mathcal{M}, q\rangle$ (or reports that $\mathcal{M}$ is empty). Here it is understood that the form q is restricted onto $\mathcal{M}$. The algorithm has runtime $O(kn)$. First we note that

$$\mathcal{M} = h \oplus \{y \in \mathcal{L}(\mathcal{K}) : (\xi, y) = \beta\},$$

where $\beta = \alpha \oplus (\xi, h)$. Let

$$S = \{a \in [k] : (\xi, g^a) = 1\}.$$

One can compute S in time $O(kn)$. If $S = \emptyset$ and $\beta = 1$ then $\mathcal{M}$ is empty. If $S = \emptyset$ and $\beta = 0$ then $\mathcal{M} = \mathcal{K}$. Otherwise pick any element $i \in S$ and remove i from S. Change the basis of $\mathcal{L}(\mathcal{K})$ according to

$$g^a \leftarrow g^a \oplus g^i \quad \text{for } a \in S.$$

Change the dual basis according to

$$\bar{g}^i \leftarrow \bar{g}^i \oplus \sum_{a\in S} \bar{g}^a.$$

Now $(g^a, \bar{g}^b) = \delta_{a,b}$ for all a, b. The basis change requires time $O(kn)$. Let us also swap the i-th and the k-th basis vectors. Updating the coefficients (D, J) using Eqs. (49,50) takes time $O(k^2) = O(kn)$. Now basis vectors $g^1, \ldots, g^{k-1}$ are orthogonal to ξ and $(\xi, g^k) = 1$. Thus

$$\mathcal{M} = h' \oplus \mathrm{span}(g^1, \ldots, g^{k-1}) \equiv h' \oplus \mathcal{L}(\mathcal{M}),$$

where $h' = h \oplus \beta g^k$ is the new shift vector. Update the coefficients (Q, D) using Eqs. (52,53), where $y = \beta g^k$. This takes time $O(k)$. Now restricting the form q onto $\mathcal{M}$ is equivalent to removing the k-th row/column from the matrix J and removing the k-th element from D. We obtained the standard form of the state $|\mathcal{M}, q\rangle$. The entire algorithm is summarized below.

function SHRINK($|\mathcal{K}, q\rangle, \xi, \alpha$)
  $S \leftarrow \{a \in [k] : (\xi, g^a) = 1\}$
  $\beta \leftarrow \alpha \oplus (\xi, h)$
  if $S = \emptyset$ and $\beta = 1$ **then**
    return EMPTY
  end if
  if $S = \emptyset$ and $\beta = 0$ **then**
    return SAME
  end if
  Pick any $i \in S$
  $S \leftarrow S \setminus \{i\}$
  for $a \in S$ **do**
    $g^a \leftarrow g^a \oplus g^i$
    Update (D, J) using Eqs. (49,50)
  end for
  $\bar{g}^i \leftarrow \bar{g}^i \oplus \sum_{a\in S} \bar{g}^a$
  Swap g^i and g^k. Swap $\bar{g}^i$ and $\bar{g}^k$.
  Update (D, J) using Eqs. (49,50)
  $h \leftarrow h \oplus \beta g^k$
  Update (Q, D) using Eqs. (52,53)
  Remove the k-th row/column from J
  Remove the k-th element from D
  $k \leftarrow k - 1$
  return SUCCESS
end function

To simplify notations, here we assume that the function SHRINK modifies the data describing the input state. The function reports whether the new affine space $\mathcal{K}$ is empty or the same as the initial space. It reports SUCESS whenever the dimension of the affine space has been reduced by one. The function has runtime $O(kn)$. Sometimes we shall use a "lazy" version of the function that does not update the coefficients of q. We shall use the notation SHRINK* for such lazy version.

APPENDIX C: THE INNER PRODUCT

Consider a pair of n-qubit stabilizer states

$$|\phi_\alpha\rangle = |\mathcal{K}_\alpha, q_\alpha\rangle, \quad \alpha = 1, 2$$

with the standard forms $(n, k_\alpha, h_\alpha, G_\alpha, \bar{G}_\alpha, Q_\alpha, D_\alpha, J_\alpha)$. Below we describe an algorithm that computes the inner product

$$\langle\phi_2|\phi_1\rangle = 2^{-(k_1+k_2)/2} \sum_{x\in\mathcal{K}_1\cap\mathcal{K}_2} e^{i\frac{\pi}{4}(q_1(x)-q_2(x))}. \quad (76)$$

in time $O(n^3)$. First we note that $x \in \mathcal{K}_2$ iff

$$x \oplus h_2 \in \mathcal{L}(\mathcal{K}_2) = \mathrm{span}(g_2^1, \ldots, g_2^{k_2}).$$

Thus $x \in \mathcal{K}_2$ iff $x \oplus h_2$ is orthogonal to all dual basis vectors $\bar{g}_2^a$ with $k_2 < a \le n$. Here and below g_α^b and $\bar{g}_\alpha^b$ denote the b-th row of G_α and $\bar{G}_\alpha$ respectively. Thus

$$\mathcal{K} \equiv \mathcal{K}_1 \cap \mathcal{K}_2 = \bigcap_{b=k_2+1}^{n} \{x \in \mathcal{K}_1 \,:\, (\bar{g}_2^b, x) = (h_2, \bar{g}_2^b)\}.$$

One can compute the standard form of $|\mathcal{K}, q_1\rangle$ by $n - k_2$ calls to the function SHRINK defined in Appendix B with $\xi = \bar{g}_2^b$ and $\alpha = (h_2, \bar{g}_2^b)$ for $b = k_2 + 1, \ldots, n$. This takes time

$$\tau_1 = O((n - k_2)k_1 n)$$

since we have to call SHRINK $n - k_2$ times.

Let $\mathcal{K} = (n, k, h, G, \bar{G})$ be the standard form of $\mathcal{K}$ and (Q_1, D_1, J_1) be the coefficients of q_1 restricted onto $\mathcal{K}$ in the basis $g^1, \ldots, g^k$ (as usual, g^a is the a-th row of G).

The next step so the compute coefficients (Q_2, D_2, J_2) of the form q_2 restricted to $\mathcal{K}$ in the basis $g^1, \ldots, g^k$ with the shift vector h. We note that

$$h = h_2 \oplus \sum_{a=1}^{k_2} y_a g_2^a, \quad \text{where} \quad y_a = (h \oplus h_2, \bar{g}_2^a).$$

One can compute $y_1, \ldots, y_{k_2}$ in time $O(k_2 n)$ and then compute the updated coefficients (Q_2, D_2) from Eqs. (52,53). This takes time $O(k_2^2)$. A simple algebra shows that $\mathcal{L}(\mathcal{K}) = \mathcal{L}(\mathcal{K}_1) \cap \mathcal{L}(\mathcal{K}_2)$, that is, $g^a \in \mathcal{L}(\mathcal{K}_2)$ for all $a = 1, \ldots, k$. Define a matrix R of size $k \times k_2$ such that

$$g^a = \sum_{b=1}^{k_2} R_{a,b} g_2^b \pmod 2, \quad 1 \le a \le k.$$

Using the dual basis of $\mathcal{K}_2$ one gets $R_{a,b} = (g^a, \bar{g}_2^b)$. One can compute the entire matrix R in time $O(kk_2 n)$. Then the coefficients (D_2, J_2) in the basis $g^1, \ldots, g^k$ can be computed from Eqs. (49,50) which takes time $O(kk_2^2)$, see Eq. (54). (Here we used a slightly stronger version of Eq. (54) taking into account that R is a rectangular matrix.) The runtime up to this point is

$$\tau_2 = \tau_1 + O(kk_2 n).$$

Now the restrictions of both forms q_1, q_2 onto $\mathcal{K}$ are defined in the same basis $g^1, \ldots, g^k$ and the same shift vector h. Thus $q \equiv q_1 - q_2$ has coefficients (Q, D, J), where $Q = Q_1 - Q_2$, $D = D_1 - D_2$, and $J = J_1 - J_2$. We get

$$\langle\phi_2|\phi_1\rangle = 2^{-(k_1+k_2)/2} \cdot W(Q, D, J),$$

where $W(Q, D, J)$ is the exponential sum Eq. (55) that can be computed in time $O(k^3)$, see Appendix A. The overall running time is thus

$$\tau = \tau_2 + O(k^3) = O((n - k_2)k_1 n + kk_2 n + k^3) = O(n^3).$$

We summarize the entire inner product algorithm below.

function InnerProduct($(|\mathcal{K}_1, q_1\rangle, |\mathcal{K}_2, q_2\rangle)$)
  $\mathcal{K} \leftarrow \mathcal{K}_1$
  for $b = k_2 + 1$ to n **do**
    $\alpha \leftarrow (h_2, \bar{g}_2^b)$
    $\epsilon \leftarrow$SHRINK$(|\mathcal{K}, q_1\rangle, \bar{g}_2^b, \alpha)$
    if $\epsilon =$EMPTY **then**
      return 0
    end if
  end for
    ▷ Now $\mathcal{K} = \mathcal{K}_1 \cap \mathcal{K}_2 = (n, k, h, G, \bar{G})$
  for $a = 1$ to k_2 **do**
    $y_a \leftarrow (h \oplus h_2, \bar{g}_2^a)$
    for $b = 1$ to k **do**
      $R_{b,a} \leftarrow (g^b, \bar{g}_2^a)$
    end for
  end for
  $h_2 \leftarrow h_2 \oplus \sum_{a=1}^{k_2} y_a g_2^a = h$
  Update (Q_2, D_2) using Eqs. (52,53) with y
  Update (D_2, J_2) using Eqs. (49,50) with R
    ▷ Now q_1, q_2 are defined in the same basis
  $Q \leftarrow Q_1 - Q_2$
  $D \leftarrow D_1 - D_2$
  $J \leftarrow J_1 - J_2$
  return $2^{-(k_1+k_2)/2}$·ExponentialSum(Q, D, J)
end function

Comments: As before, we assume that the output is converted to a triple of integers (ϵ, p, m) such that $\langle\phi_2|\phi_1\rangle = \epsilon \cdot 2^{p/2} \cdot e^{i\pi m/4}$. If both k_1 and k_2 are small, one can compute the intersection $\mathcal{K}_1 \cap \mathcal{K}_2$ directly by solving a linear system

$$\sum_{a=1}^{k_1} x_a g_1^a \oplus \sum_{b=1}^{k_2} y_b g_2^b = h_1 \oplus h_2$$

with $k_1 + k_2$ variables and n equations. This provides a shift vector and a basis for $\mathcal{K}$ in time $O(n(k_1 + k_2)^2)$. Then one can compute the updated coefficients of q_1 and q_2 in the new basis in time $O(k(k_1^2 + k_2^2))$. Thus the overall running time is

$$\tau = O(k_1^2 n + k_2^2 n + k^3) \quad (77)$$

which is linear in n provided that both $k_1, k_2 = O(1)$. We note however that the vast majority of stabilizer states have $k_\alpha \approx n$, see Appendix D, so the above method provides no speedup in the generic case.

The timing analysis of the function InnerProduct reported in Table 1 was performed for inner products $\langle\tilde{x}|\phi\rangle$, where $\phi \in \mathcal{S}_n$ is drawn from the uniform distribution (as described in Appendix D), $x \in \mathbb{F}_2^n$ is a random uniformly

distributed string, and $|\tilde{x}\rangle \equiv |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \cdots \otimes \tilde{x}_n\rangle$, where $|\tilde{0}\rangle = |0\rangle$ and $|\tilde{1}\rangle = H|0\rangle$. This choice is justified since our simulation algorithm only requires inner products of the above form.

APPENDIX D: RANDOM STABILIZER STATES

Let us now describe an algorithm that generates a random uniformly distributed stabilizer state $|\mathcal{K}, q\rangle \in \mathcal{S}_n$. The algorithm has average-case runtime $O(n^2)$ and the worst-case runtime $O(n^3)$.

For each $0 \le k \le n$ define a subset of stabilizer states

$$\mathcal{S}_n^k = \{|\mathcal{K}, q\rangle \in \mathcal{S}_n \,:\, \dim(\mathcal{K}) = k\}.$$

For example, $\mathcal{S}_n^0$ includes all basis vectors, whereas $\mathcal{S}_n^n$ includes stabilizer states supported on all basis vectors. Our algorithm first picks a random integer $d = 0, 1, \ldots, n$ drawn from a distribution

$$P(d) = \frac{|\mathcal{S}_n^{n-d}|}{\sum_{m=0}^n |\mathcal{S}_n^m|} \tag{78}$$

and generates a random subspace $\mathcal{K} \subseteq \mathbb{F}_2^n$ of dimension $k = n - d$. To compute $P(d)$ we need the following fact.

Lemma 5.

$$|\mathcal{S}_n^{n-d}| = 8 \cdot 2^{n+\frac{1}{2}[n(n+1)-d(d+1)]} \cdot \prod_{a=1}^d \frac{1-2^{d-n-a}}{1-2^{-a}}. \tag{79}$$

for any $d = 1, \ldots, n$ *and* $|\mathcal{S}_n^n| = 8 \cdot 2^{n+\frac{1}{2}n(n+1)}$.

Proof. Let $k \equiv n-d$. The number of k-dimensional linear subspaces $\mathcal{L} \subseteq \mathbb{F}_2^n$ is known to be

$$\Gamma_n^k = \Gamma_n^d = \prod_{m=0}^{d-1} \frac{2^n - 2^m}{2^d - 2^m}$$

For a given $\mathcal{L}$ there are 2^{n-k} affine spaces $\mathcal{K}$ such that $\mathcal{K} = \mathcal{L} \oplus h$ for some shift vector h. Finally, for a given affine space $\mathcal{K}$ there are

$$\Lambda_n^k = 8 \cdot 2^{2k} \cdot 2^{k(k-1)/2}$$

quadratic forms $q : \mathcal{K} \to \mathbb{Z}_8$. Here the three factors represent the number of choices for the coefficients (Q, D, J) in Eqs. (43,44,45) respectively (recall that the diagonal of J is determined by D, see Eq. (46)). It follows that $|\mathcal{S}_n^k| = 2^{n-k} \cdot \Gamma_n^k \cdot \Lambda_n^k$, which gives Eq. (79). □

One can rewrite Eq. (78) as

$$P(d) = \frac{\eta(d)}{\sum_{m=0}^n \eta(m)}, \tag{80}$$

where $\eta(0) = 1$ and

$$\eta(d) = 2^{-d(d+1)/2} \cdot \prod_{a=1}^d \frac{1-2^{d-n-a}}{1-2^{-a}}$$

for $d = 1, \ldots, n$. One can compute a lookup table for the function $\eta(d)$ offline since it depends only on n. Clearly, $d = O(1)$ with high probability. Thus, the average-case online complexity of sampling d from the distribution $P(d)$ is $O(1)$.

We start by choosing the zero shift vector such that $\mathcal{K}$ is a random linear space of dimension k. We shall generate $\mathcal{K}$ by repeatedly picking a random matrix $X \in \mathbb{F}_2^{d\times n}$ until X has rank d and then choosing $\mathcal{K} = \ker(X)$. It is well-known that X has rank d with probability

$$p_{n,d} = \prod_{a=0}^{d-1} (1 - 2^{-n+a}) \ge \max\{1/4, 1 - 2^{-n+d}\}.$$

Note that $p_{n,d}$ is exponentially close to 1 whenever $d = O(1)$. Thus X has full rank after $O(1)$ attempts with high probability. Furthermore, one can compute the rank of X in time $O(nd^2)$ using the Gaussian elimination by bringing X into the row echelon form. It is also well-known that conditioned on X having full rank, the subspace $\ker(X)$ is distributed uniformly on the set of all subspaces of $\mathbb{F}_2^n$ of dimension $n-d$. Thus we can choose $\mathcal{K} = \ker(X)$.

The next step is computing $n \times n$ matrices G and $\bar{G}$ such that $\mathcal{K}$ is spanned by the first k rows of G and $G\bar{G}^T = I$. Let us first set $\mathcal{K} = \mathbb{F}_2^n$ and $G = \bar{G} = I$. Choose a zero quadratic form $q(x) = 0$ for all $x \in \mathcal{K}$. Let ξ^a be the a-th row of the matrix X. One can make $\mathcal{K}$ orthogonal to $\xi^1, \ldots, \xi^d$ by making d calls to the function $\mathrm{SHRINK}^*(|\mathcal{K}, q\rangle, \xi^a, 0)$ defined in Appendix B. (Recall that SHRINK^* does not update the coefficients of q.) Finally we shift $\mathcal{K}$ by a random uniformly distributed vector $h \in \mathbb{F}_2^n$. At this point $\mathcal{K}$ is a random affine space represented in the standard form. It remains to choose random coefficients of the quadratic form $q : \mathcal{K} \to \mathbb{Z}_8$ in the basis $g^1, \ldots, g^k$. Since q must be distributed uniformly on the set of all quadratic forms $q : \mathcal{K} \to \mathbb{Z}_8$, we must choose $Q \in \mathbb{Z}_8$, $D_a \in \{0, 2, 4, 6\}$, and $J_{a,b} \in \{0, 4\}$ for $a < b$ as random uniform elements of the respective sets. Then the entire matrix J is determined by $J_{b,a} = J_{a,b}$ and $J_{a,a} = 2D_{a,a}$, see Eq. (46). The entire algorithm is summarized below.

function RANDOMSTABILIZERSTATE(n)
  Compute $P(0),\dots,P(n)$ from Eq. (80)
  Sample $d \in \{0,1,\dots,n\}$ from $P(d)$
  $k \leftarrow n-d$
  repeat
    Pick random $X \in \mathbb{F}_2^{d\times n}$
  until $\mathrm{rank}(X) = d$
  $G \leftarrow I$, $\bar{G} \leftarrow I$, $h \leftarrow 0^k$
  $\mathcal{K} \leftarrow (n,k,h,G,\bar{G})$
    ▷ Now $\mathcal{K} = \mathbb{F}_2^n$ is full binary space
  $q \leftarrow$ all-zeros function on $\mathcal{K}$
  for $a = 1$ to d **do**
    $\xi \leftarrow a$-th row of X
    SHRINK$^*(|\mathcal{K},q\rangle, \xi, 0)$
  end for
    ▷ Now $\mathcal{K} = \ker(X)$
    ▷ $\mathcal{K}$ has the standard form
  Pick random $h \in \mathbb{F}_2^n$
  Pick random $Q \in \mathbb{Z}_8$
  Pick random $D_a \in \{0,2,4,6\}$
  Pick random $J_{a,b} = J_{b,a} \in \{0,4\}$ for $a \neq b$
  Set $J_{a,a} = 2D_a \pmod 8$
  return $(n,k,h,G,\bar{G},Q,D,J)$
end function

Each call to SHRINK takes time $O(n^2)$, see Appendix B, whereas each computation of $\mathrm{rank}(X)$ takes time $O(dn^2)$. Thus the entire algorithm takes time $O(dn^2)$. Since $d = O(1)$ with high probability, see above, the average runtime is $O(n^2)$, whereas the worst-case runtime is $O(n^3)$. Timing analysis for a MATLAB implementation is reported in Table 1.

APPENDIX E: PAULI MEASUREMENTS

Suppose $|\mathcal{K},q\rangle \in \mathcal{S}_n$ is a stabilizer state of n qubits represented in the standard form and $P \in \mathcal{P}_n$ is a Pauli operator. Define an operator

$$P_+ \equiv \frac{1}{2}(I+P).$$

It is well-known that P_+ maps stabilizer states to (unnormalized) stabilizer states. Note that P_+ is a projector if P is self-adjoint and $\sqrt{2}P_+$ is a unitary Clifford operator if $P^\dagger = -P$. Below we describe an algorithm that computes the normalization and the standard form of the state $P_+|\mathcal{K},q\rangle$. The algorithm has runtime $O(n^2)$. We shall be mostly interested in the case when P_+ is a projector (although our algorithm applies to the general case). Note that a projector onto the codespace of any stabilizer code with a stabilizer group $\mathcal{G} \subseteq \mathcal{P}_n$ can be written as a product of at most n projectors P_+ associated with some set of generators of $\mathcal{G}$. Thus a projected state $\Pi_{\mathcal{G}}|\mathcal{K},q\rangle$ can be computed in time $O(n^3)$ using the above algorithm.

Let $\mathcal{K} = (n,k,h,G,\bar{G})$ be the standard form of $\mathcal{K}$ and

$$P = i^m Z(\zeta)X(\xi), \quad m \in \mathbb{Z}_4, \quad \xi,\zeta \in \mathbb{F}_2^n. \tag{81}$$

We shall consider two cases depending on whether or not $\xi \in \mathcal{L}(\mathcal{K})$. This inclusion can be checked in time $O(kn)$ by computing inner products $\xi_a = (\xi, \bar{g}^a)$ with $a = 1,\dots,k$. Namely, $\xi \in \mathcal{L}(\mathcal{K})$ iff $\xi = \sum_{a=1}^k \xi_a g^a \pmod 2$.

Case 1: $\xi \in \mathcal{L}(\mathcal{K})$. Define a function

$$\chi(x) = q(x\oplus\xi) - q(x). \tag{82}$$

By definition of a quadratic form one has

$$\chi(h\oplus y) = \chi(h) + J(\xi,y) \quad \text{for all } y \in \mathcal{L}(\mathcal{K}). \tag{83}$$

The state $P_+|\mathcal{K},q\rangle$ can be written as

$$2^{-k/2-1}\sum_{x\in\mathcal{K}} e^{i\frac{\pi}{4}q(x)}\left(1 + i^m(-1)^{(\zeta,x)}e^{i\frac{\pi}{4}\chi(x)}\right)|x\rangle. \tag{84}$$

Perform a change of variable $x = h\oplus y$ with $y \in \mathcal{L}(\mathcal{K})$. Using Eq. (83) one can rewrite the above state as

$$2^{-k/2-1}\sum_{y\in\mathcal{L}(\mathcal{K})} e^{i\frac{\pi}{4}q(h\oplus y)}\left(1 + e^{i\frac{\pi}{4}(\omega+\lambda(y))}\right)|h\oplus y\rangle, \tag{85}$$

with

$$\omega = 2m + 4(\zeta,h) + q(h\oplus\xi) - q(h) \in \{0,2,4,6\} \tag{86}$$

and

$$\lambda(y) = 4(\zeta,y) + J(\xi,y) \in \{0,4\}. \tag{87}$$

Let us first compute ω. We have

$$\xi = \sum_{a=1}^k \xi_a g^a \pmod 2, \quad \xi_a = (\bar{g}^a,\xi). \tag{88}$$

The decomposition Eq. (88) can be computed in time $O(kn)$. Once the coefficients ξ_a are known, one can compute ω from

$$\omega = 2m + 4(\zeta,h) + \sum_{a=1}^k D_a\xi_a + \sum_{1\le a<b\le k} J_{a,b}\xi_a\xi_b. \tag{89}$$

This takes time $O(kn)$.

Suppose first that $\omega \in \{0,4\}$. Then $e^{i\frac{\pi}{4}\omega} = \pm 1$ and thus

$$1 + e^{i\frac{\pi}{4}(\omega+\lambda(y))} = \begin{cases} 2 & \text{if } \lambda(y)+\omega = 0 \pmod 8 \\ 0 & \text{if } \lambda(y)+\omega = 4 \pmod 8. \end{cases}$$

We get

$$P_+|\mathcal{K},q\rangle = 2^{-k/2}\sum_{x\in\mathcal{M}} e^{i\frac{\pi}{4}q(x)}|x\rangle. \tag{90}$$

where

$$\mathcal{M} = \mathcal{K} \cap \{x \in \mathbb{F}_2^n \,:\, \lambda(h\oplus x) = \omega\}. \tag{91}$$

Let us choose a vector $\gamma \in \mathbb{F}_2^n$ such that $\lambda(y) = 4(\gamma, y)$ for all $y \in \mathcal{L}(\mathcal{K})$. We shall look for

$$\gamma = \sum_{b=1}^{k} \eta_b \bar{g}^b \pmod 2, \quad \eta_b \in \{0,1\}. \tag{92}$$

Choosing $y = g^a$ and using $(g^a, \bar{g}^b) = \delta_{a,b}$ one gets

$$4\eta_a = \lambda(g^a) = 4(\zeta, g^a) + J(\xi, g^a), \quad 1 \le a \le k. \tag{93}$$

To compute (ζ, g^a) and $J(\xi, g^a)$ consider expansions Eq. (88) and

$$\zeta = \sum_{a=1}^{n} \zeta_a \bar{g}^a \pmod 2, \quad \zeta_a = (g^a, \zeta). \tag{94}$$

One can compute all the coefficients $\zeta_1, \ldots, \zeta_k$ in time $O(kn)$. The fact that $J(x, y)$ is a bilinear form implies

$$4\eta_a = 4\zeta_a + \sum_{b=1}^{k} J_{a,b}\xi_b, \quad 1 \le a \le k. \tag{95}$$

Thus $\eta_1, \ldots, \eta_k$ can be computed in time $O(kn)$. Let $\omega = 4\omega'$ with $\omega' \in \{0,1\}$. We arrived at

$$\mathcal{M} = \mathcal{K} \cap \{x \in \mathbb{F}_2^n \,:\, (\gamma, x) = \alpha\}, \quad \alpha \equiv \omega' \oplus (\gamma, h).$$

The standard form of the state defined in Eq. (90) can be computed by calling the function SHRINK$(|\mathcal{K}, q\rangle, \gamma, \alpha)$, see Appendix B, which takes time $O(kn)$.

Next suppose that $\omega \in \{2, 6\}$. Then $e^{i\frac{\pi}{4}\omega} = \pm i$ and thus

$$1 + e^{i\frac{\pi}{4}(\omega+\lambda(y))} = \begin{cases} \sqrt{2}e^{i\frac{\pi}{4}} & \text{if } \lambda(y) + \omega = 2 \pmod 8 \\ \sqrt{2}e^{-i\frac{\pi}{4}} & \text{if } \lambda(y) + \omega = 6 \pmod 8 \end{cases}$$

We shall choose a quadratic form $\lambda' : \mathcal{K} \to \mathbb{Z}_8$ such that

$$\lambda'(h \oplus y) = \begin{cases} 0 & \text{if } \lambda(y) = 0, \\ 2 & \text{if } \lambda(y) = 4. \end{cases} \tag{96}$$

Define

$$\sigma = \begin{cases} 1 & \text{if } \omega = 2, \\ -1 & \text{if } \omega = 6. \end{cases} \tag{97}$$

Then the state in Eq. (85) can be written as

$$P_+|\mathcal{K}, q\rangle = 2^{-(k+1)/2} \sum_{x \in \mathcal{K}} e^{i\frac{\pi}{4}q'(x)}|x\rangle = 2^{-1/2}|\mathcal{K}, q'\rangle \tag{98}$$

with a quadratic form

$$q'(x) = \sigma + q(x) - \sigma\lambda'(x). \tag{99}$$

To get the standard form of $|\mathcal{K}, q'\rangle$ we need to choose $\lambda'(x)$ satisfying Eq. (96) and compute the coefficients of $\lambda'(x)$ in the basis $g^1, \ldots, g^k$ of $\mathcal{L}(\mathcal{K})$. First, let us compute the basis-dependent representation of $\lambda(y)$. Suppose $y = \sum_{a=1}^{k} y_a g^a \pmod 2$ and let $\vec{y} = (y_1, \ldots, y_k)$. Substituting Eqs. (88,94) into Eq. (87) one gets

$$\lambda(\vec{y}) = 4\sum_{a=1}^{k} \eta_a y_a,$$

where $\eta_a \in \{0,1\}$ are defined by Eq. (95). For any $z_1, \ldots, z_k \in \{0,1\}$ one has the following identity:

$$2(z_1 \oplus \cdots \oplus z_k) = 2\sum_{a=1}^{k} z_a - 4 \sum_{1 \le a < b \le k} z_a z_b \pmod 8.$$

Choose $z_a = \eta_a y_a$ such that $\lambda(\vec{y}) = 4(z_1 \oplus \cdots \oplus z_k)$. Then a function $\lambda'(y)$ satisfying Eq. (96) has a basis-dependent representation $\lambda'(\vec{y}) = 2(z_1 \oplus \cdots \oplus z_k)$, that is,

$$\lambda'(h \oplus y) = 2\sum_{a=1}^{k} \eta_a y_a - 4 \sum_{1 \le a < b \le k} \eta_a \eta_b \, y_b y_b. \tag{100}$$

To summarize, the coefficients of the form q' in the basis $g^1, \ldots, g^k$ are (Q', D', J'), where

$$Q' = Q + \sigma, \quad D'_a = D_a - 2\sigma\eta_a, \tag{101}$$

and

$$J'_{a,b} = J_{a,b} + 4\eta_a\eta_b \quad \text{for } a \ne b. \tag{102}$$

This determines the standard form of $|\mathcal{K}, q'\rangle$.

Case 2: $\xi \notin \mathcal{L}(\mathcal{K})$. Then $\xi \oplus x \notin \mathcal{K}$ for any $x \in \mathcal{K}$ and thus the states $|\mathcal{K}, q\rangle$ and $P|\mathcal{K}, q\rangle$ are supported on disjoint subsets of basis vectors. Define an affine space $\mathcal{M} = \mathcal{L}(\mathcal{M}) \oplus h$ of dimension $k+1$, where $\mathcal{L}(\mathcal{M})$ is spanned by $\mathcal{L}(\mathcal{K})$ and ξ. We equip $\mathcal{L}(\mathcal{M})$ with a basis $g^1, \ldots, g^{k+1}$, where $g^{k+1} \equiv \xi$. Then any vector $x \in \mathcal{M}$ can be written in a basis-dependent way as

$$x = h \oplus \sum_{a=1}^{k+1} x_a g^a \pmod 2.$$

Let $\vec{x} = (x_1, \ldots, x_{k+1})$. A simple algebra shows that

$$P_+|\mathcal{K}, q\rangle = 2^{-1-k/2} \sum_{x \in \mathcal{M}} e^{i\frac{\pi}{4}q'(x)}|x\rangle = 2^{-1/2}|\mathcal{M}, q'\rangle, \tag{103}$$

where $q' : \mathcal{M} \to \mathbb{Z}_8$ is a quadratic form defined by

$$q'(\vec{x}) = q(\vec{x}) + [2m + 4(\zeta, h \oplus \xi)]\, x_{k+1} + 4\sum_{a=1}^{k} \zeta_a x_a x_{k+1}. \tag{104}$$

Here it is understood that $q(\vec{x})$ depends only on the first k coordinates of x. Thus the coefficients of q' in the chosen basis of $\mathcal{L}(\mathcal{M})$ are $Q' = Q$, $D' = [D, 2m + 4(\zeta, h \oplus \xi)]$, and

$$J' = \left[\begin{array}{c|c} J & 4\zeta^T \\ \hline 4\zeta & 4m \end{array}\right]. \tag{105}$$

Here $\boldsymbol{\zeta} \equiv (\zeta_1, \ldots, \zeta_k)$ is a row vector.

It remains to compute the standard form of $\mathcal{M}$. Below we define a function EXTEND$(\mathcal{K}, \xi)$ that takes as input an affine space $\mathcal{K} = (n, k, h, G, \bar{G})$ and a vector $\xi \in \mathbb{F}_2^n$. If $\xi \in \mathcal{L}(\mathcal{K})$, the function does nothing. Otherwise, the function outputs an affine space $\mathcal{M} = (n, k+1, h, H, \bar{H})$ such that the first k rows of G and H are the same and the $(k+1)$-th row of H equals ξ. Since the function EXTEND is very similar to the function SHRINK defined in Appendix B, we just state the algorithm skipping the analysis.

function $\mathcal{M}$=EXTEND$(\mathcal{K}, \xi)$
  $S \leftarrow \{a \in [n] : (\xi, \bar{g}^a) = 1\}$
  $T \leftarrow S \cap \{k+1, \ldots, n-1, n\}$
  if $T = \emptyset$ **then**
    ▷ $\xi \in \mathcal{L}(\mathcal{K})$
    return $\mathcal{K}$
  end if
  Pick any $i \in T$
  $S \leftarrow S \setminus \{i\}$
  for $a \in S$ **do**
    $\bar{g}^a \leftarrow \bar{g}^a \oplus \bar{g}^i$
  end for
  $g^i \leftarrow g^i \oplus \sum_{a \in S} g^a$
    ▷ Now $g^i = \xi$
  Swap g^i and g^{k+1}. Swap $\bar{g}^i$ and $\bar{g}^{k+1}$.
  return $(n, k+1, h, G, \bar{G})$
end function

It has runtime $O(n^2)$. We do not have to update the coefficients of q' since Eq. (104) defines q' in the basis $g^1, \ldots, g^k, \xi$ which coincides with the new basis of $\mathcal{M}$. We conclude that the projected state $2^{-1/2}|\mathcal{M}, q'\rangle$ can be computed in time $O(n^2)$.

Below we summarize the entire algorithm as a function MeasurePauli that takes as input a stabilizer state $|\mathcal{K}, q\rangle \in \mathcal{S}_n$ and a Pauli operator $P \in \mathcal{P}_n$. The function returns the norm of the projected state $\Gamma = \|P_+|\mathcal{K}, q\rangle\|$. If $\Gamma \neq 0$, the function computes the standard form of the projected state $P_+|\mathcal{K}, q\rangle$. As before, we assume that the function can modify the data describing the input state.

function Γ=MEASUREPAULI$(|\mathcal{K}, q\rangle, P)$
    ▷ $P = i^m Z(\zeta) X(\xi)$
    ▷ $\mathcal{K} = (n, k, h, G, \bar{G})$
    ▷ $q = (Q, D, J)$
  for $a = 1$ to k **do**
    $\xi_a \leftarrow (\bar{g}^a, \xi)$, $\zeta_a \leftarrow (g^a, \zeta)$
  end for
  $\xi' \leftarrow \sum_{a=1}^k \xi_a g^a \pmod 2$
  Compute $\omega \in \{0, 2, 4, 6\}$ using Eq. (89)
  if $\xi' = \xi$ and $\omega \in \{0, 4\}$ **then**
    Compute $\eta_1, \ldots, \eta_k$ using Eq. (95)
    $\gamma \leftarrow \sum_{a=1}^k \eta_a g^a \pmod 2$
    $\omega' \leftarrow \omega/4$
    $\alpha \leftarrow \omega' \oplus (\eta, h)$
    $\epsilon \leftarrow$SHRINK$(|\mathcal{K}, q\rangle, \gamma, \alpha)$
    if $\epsilon =$EMPTY **then**
      $\Gamma \leftarrow 0$
      return
    end if
    if $\epsilon =$SAME **then**
      $\Gamma \leftarrow 1$
      return
    end if
    if $\epsilon =$SUCCESS **then**
      $\Gamma \leftarrow 2^{-1/2}$
      return
    end if
  end if
  if $\xi' = \xi$ and $\omega \in \{2, 6\}$ **then**
    $\sigma \leftarrow 2 - (\omega/2)$
    Compute (Q', D', J') using Eqs. (101,102)
    $(Q, D, J) \leftarrow (Q', D', J')$
    $\Gamma \leftarrow 2^{-1/2}$
    return
  end if
  if $\xi' \neq \xi$ **then**
    $\mathcal{K} \leftarrow$EXTEND$(\mathcal{K}, \xi)$
    $D \leftarrow [D, 2m + 4(\zeta, h \oplus \xi)]$
    $J \leftarrow J'$, where J' is defined in Eq. (105)
    $\Gamma \leftarrow 2^{-1/2}$
    return
  end if
end function

Number of qubits	**10**	**25**	**50**	**75**	**100**	n
MeasurePauli	0.27	0.3	0.4	0.5	0.6	$O(n^2)$
RandomStabilizerState	0.2	0.3	0.8	1.7	2.8	$O(n^2)$
InnerProduct	0.5	1.5	3.5	6.5	8.9	$O(n^3)$
ExponentialSum	0.3	0.8	2.2	4.4	8	$O(n^3)$

TABLE I. Average runtime in milliseconds for a MATLAB implementation of our algorithms. Simulations were performed on a laptop with 2.6GHz Intel i5 Dual Core CPU.

APPENDIX F: SIMULATION OF THE HIDDEN SHIFT ALGORITHM

Here we provide further details of the simulations reported in Fig. 1. Recall that we simulate a circuit

$$U = H^{\otimes n} O_{f'} H^{\otimes n} O_f H^{\otimes n}, \tag{106}$$

where $O_f|x\rangle = f(x)|x\rangle$ and $O_{f'}|x\rangle = f'(x)|x\rangle$ are oracle circuits for some bent functions $f, f' : \mathbb{F}_2^n \to \{+1, -1\}$ such that

$$f'(x \oplus s) = 2^{-n/2} \sum_{y \in \mathbb{F}_2^n} (-1)^{x \cdot y} f(y) \quad \text{for all } x \in \mathbb{F}_2^n. \tag{107}$$

Here $s \in \mathbb{F}_2^n$ is the hidden shift that can be found from $|s\rangle = U|0^n\rangle$. In our simulations the hidden shift s was chosen at random from the uniform distribution. The function f was chosen from (a subclass of) the Maiorana McFarland family of bent functions. In general, a Maiorana McFarland bent function is defined as follows. Suppose n is even. Let

$$g : \mathbb{F}_2^{n/2} \to \mathbb{F}_2 \quad \text{and} \quad \pi : \mathbb{F}_2^{n/2} \to \mathbb{F}_2^{n/2}$$

be any Boolean function and any permutation respectively. For any such pair g, π we may define a bent function $f : \mathbb{F}_2^n \to \{+1, -1\}$ according to

$$f(x, y) = (-1)^{g(x) + y \cdot \pi(x)} \qquad x, y \in \mathbb{F}_2^{n/2}. \tag{108}$$

The Hadamard transform of f is given by

$$2^{-n/2} \sum_{u,v} (-1)^{u \cdot x + v \cdot y} f(u, v) = (-1)^{x \cdot \pi^{-1}(y) + g(\pi^{-1}(y))}. \tag{109}$$

In our simulations we only used bent functions of the form Eq. (108) with $\pi = I$ (the identity permutation). The Boolean function g was chosen at random, as explained below. Letting O_g be the $n/2$-qubit diagonal unitary

$$O_g|x\rangle = (-1)^{g(x)}|x\rangle \quad x \in \mathbb{F}_2^{n/2}$$

we see that a quantum circuit which implements the n-qubit unitary oracle $O_f|x, y\rangle = f(x, y)|x, y\rangle$ can be decomposed as

$$O_f = \left(\prod_{i=1}^{n/2} CZ_{i,i+n/2} \right) O_g \otimes I$$

where $CZ = \mathrm{diag}(1, 1, 1, -1)$ is the two-qubit controlled-Z gate. Here the tensor product separates the first $n/2$ qubits from the last $n/2$. Likewise, from Eqs. (107,109) one infers that

$$O_{f'} = X(s) \left[\left(\prod_{i=1}^{n/2} CZ_{i,i+n/2} \right) I \otimes O_g \right] X(s)$$

Note that the total T-count of the circuit U is twice the T-count of O_g. To construct a circuit implementing O_g we chose a sequence of gates from the set $\{Z, CZ, CCZ\}$, where CCZ is the controlled-controlled-Z gate. We first fixed the number of CCZ gates (five and six for the simulations reported in the left/right plots of Fig. 1 respectively), and then produced a circuit O_g alternating the CCZ gates (on a randomly chosen triple of qubits) with random sequences of 200 Clifford gates from the set $\{Z, CZ\}$. Note that the CCZ gate can be replaced by the Toffoli gate using the identity

$$CCZ = (I \otimes I \otimes H)\mathrm{Toff}(I \otimes I \otimes H). \tag{110}$$

To decompose Toffoli gates into Clifford and T-gates we used a gadget proposed by Jones [13], see Fig. 3. The gadget uses four T-gates, two ancillary qubits initialized in the state $|0\rangle$, several Clifford gates, and the $0, 1$-measurement. The final Clifford gate is classically controlled by the measurement outcome. To simulate the gadget we use the trick described in the remark between Eqs. (29, 30). Namely, in our simulation the measurement of the ancillary qubit is replaced by postselection on a random output bit y in exactly the same way as was done for the T-gate gadget. The second ancilla in the gadget is never measured and is returned to the state $|0\rangle$ at the output; this ancilla is reused by all Toffolis in the circuit.

The simulation algorithm we implemented differs in some small details from the algorithm analyzed in the main text of the paper. To produce each data point in Fig. 1 we first fixed the output qubit $q \in \{1, 2, \ldots, 40\}$. We then estimated the ratio (cf. Eq. (27))

$$P_{out}^y(1) = \frac{\langle 0^N \otimes \psi | V_y^\dagger (|1\rangle\langle 1|_q \otimes |y\rangle\langle y|) V_y | 0^N \otimes \psi \rangle}{\langle 0^N \otimes \psi | V_y^\dagger (I_{n+1} \otimes |y\rangle\langle y|) V_y | 0^N \otimes \psi \rangle}. \tag{111}$$

for a randomly chosen postselection bit-string y. For us $N = n + n_{anc}$ where $n = 40$ is the number of qubits in the original circuit to be simulated while n_{anc} is the number of ancillae initialized in the state $|0\rangle$ which are used for the Toffoli gadgets. Since each Toffoli gadget requires two ancilla, one of which is shared by all of them, we have $n_{anc} = 1 + \mathrm{TF}$ where TF is the number of Toffoli gadgets used. In Eq. (111) the number of postselection bits is $|y| = t + \mathrm{TF}$ where t is the number of T-gates in the circuit (including the four T-gates within each Toffoli gadget). The unitary V_y is a $(n + 1 + \mathrm{TF} + t)$-qubit Clifford unitary which is obtained by replacing all Toffoli gadgets and T gate gadgets by the appropriate Clifford circuits obtained by postselecting on the measurement outcomes defined by the bit string y. Finally, the state ψ in Eq. (111) is a t-qubit state which approximates t copies of the magic state $|A\rangle^{\otimes t}$. In particular, ψ was derived from a k-dimensional subspace $\mathcal{L}$ of $\mathbb{F}_2^t$ in the manner described in the main text of the paper. In our simulations we used $k = 11$ (left plot in Fig. 1) and $k = 12$ (right plot in Fig. 1). The fidelities were $|\langle A^{\otimes t}|\psi\rangle| \approx 0.81$ and $|\langle A^{\otimes t}|\psi\rangle| \approx 0.69$ respectively.

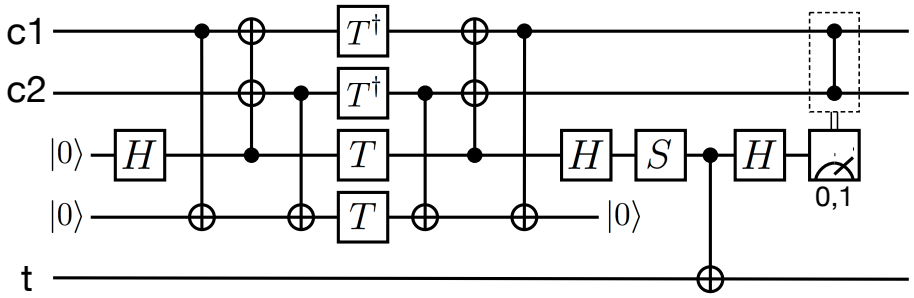

FIG. 3. Gadget from Ref. [13] implementing the Toffoli gate. The two control qubits and the target qubit are denoted c_1, c_2 and t respectively. Both measurement outcomes appear with probability 1/2. The final controlled-Z gate on qubits c_1, c_2 is applied only if the measurement outcome is '1'.

To estimate $P^y_{out}(1)$ we computed integers u, v and stabilizer groups $\mathcal{F}, \mathcal{G}$ such that

$$\langle 0^N \otimes \psi | V_y^\dagger (|1\rangle\langle 1|_q \otimes |y\rangle\langle y|) V_y | 0^N \otimes \psi \rangle = 2^{-u} \langle \psi | \Pi_{\mathcal{F}} | \psi \rangle \tag{112}$$

and

$$\langle 0^N \otimes \psi | V_y^\dagger (|0\rangle\langle 0|_q \otimes |y\rangle\langle y|) V_y | 0^N \otimes \psi \rangle = 2^{-v} \langle \psi | \Pi_{\mathcal{G}} | \psi \rangle \tag{113}$$

and then, if $\Pi_{\mathcal{F}} \neq 0$ and $\Pi_{\mathcal{G}} \neq 0$, we computed approximations α, β to the quantities Eqs. (113,112) using the norm estimation procedure described in the main text. The number of random stabilizer states sampled by the norm estimation procedure was chosen to be 100 (left plot in Fig. 1) or 50 (right plot in Fig. 1). Our estimate of $P^y_{out}(1)$ was then $\alpha/(\alpha+\beta)$ (cf. Eq. (111)). Note that if either $\Pi_{\mathcal{F}} = 0$ or $\Pi_{\mathcal{G}} = 0$ then α, β, and $P^y_{out}(1)$ can be computed without ever calling the norm estimation subroutine. This special case occured for all qubits $1, 2, \ldots, 20$ in both our simulations (as well as for some of the other data points).

[1] D. Wecker and K. M. Svore, preprint arXiv:1402.4467 (2014).
[2] D. Gottesman, preprint quant-ph/9807006 (1998).
[3] S. Aaronson and D. Gottesman, Phys. Rev. A **70**, 052328 (2004).
[4] I. Markov and Y. Shi, SIAM J. on Comp. **38**, 963 (2008).
[5] M. Van den Nest, Quant. Inf. Comp. **10**, 0258 (2010).
[6] H. Pashayan, J. Wallman, and S. Bartlett, preprint arXiv:1503.07525 (2015).
[7] S. Bravyi and A. Kitaev, preprint quant-ph/9811052 (1998).
[8] A. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).
[9] V. Kliuchnikov, D. Maslov, and M. Mosca, Quant. Inf. and Comp. **13**, 607 (2013).
[10] P. Selinger, Quant. Inf. and Comp. **15**, 159 (2015).
[11] N. J. Ross and P. Selinger, preprint arXiv:1403.2975 (2014).
[12] A. Fowler, S. Devitt, and C. Jones, Scientific Reports **3**, 1939 (2013).
[13] C. Jones, Physical Review A **87**, 022328 (2013).
[14] Note1, The MATLAB implementation of the sampling algorithm is available upon request to the authors.
[15] S. Bravyi, G. Smith, and J. Smolin, preprint arXiv:1506.01396 (2015).
[16] M. Rötteler, in *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms* (2010), pp. 448–457.
[17] C. Dankert, R. Cleve, J. Emerson, and E. Livine, Phys. Rev. A **80**, 012304 (2009).
[18] H. J. García, I. Markov, and A. Cross, Quant. Inf. and Comp. **14**, 683 (2014).
[19] X. Zhou, D. W. Leung, and I. L. Chuang, Phys. Rev. A **62**, 052316 (2000).
[20] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani, Theoretical Computer Science **43**, 169 (1986).
[21] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **32**, 818 (2013), 1206.0758.
[22] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, Quant. Inf. and Comp. **14**, 1261 (2014).
[23] A. Kitaev, private communication (2003).
[24] K.-U. Schmidt, Information Theory, IEEE Transactions on **55**, 5803 (2009).
[25] M. Araújo (2011), URL http://www.math.ist.utl.pt/~ggranja/manuel.pdf.
[26] J. Dehaene and B. De Moor, Phys. Rev. A **68**, 042318 (2003).
[27] H. García-Ramírez, Ph.D. thesis, The University of Michigan (2014).