Improved classical simulation of quantum circuits dominated by Clifford gates: 
Supplemental Material

Sergey Bravyi\textsuperscript{1} and David Gosset\textsuperscript{2}
\textsuperscript{1}IBM T.J. Watson Research Center, Yorktown Heights NY 10598
\textsuperscript{2}Walter Burke Institute for Theoretical Physics and Institute for Quantum Information and Matter, California Institute of Technology

I. STABILIZER FORMALISM

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

\[
\Pi_G = 2^{-r} \sum_{P \in G} P. \tag{1}
\]

A pure \( n \)-qubit state \( |\psi\rangle \) is a stabilizer state iff \( |\psi\rangle = U|0^\otimes n\rangle \) for some Clifford unitary \( U \). Any stabilizer state \( |\psi\rangle \) is uniquely defined (up to the overall phase) by a stabilizer group \( G \subseteq P_n \) of dimension \( n \) such that \( |\psi\rangle \) is the only state stabilized by \( G \). Let \( \mathcal{S}_n \) be the set of all \( n \)-qubit stabilizer states. This set is known to be a 2-design [1], that is,

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

where the integral is with respect to the Haar measure on the set of normalized \( n \)-qubit states \( |\phi\rangle \).

Throughout the paper we assume that stabilizer states are represented in a certain standard form defined in Section IV B. In this representation, three basic tasks can be performed efficiently. First, one can compute the inner product between stabilizer states [2–4]. More precisely, consider stabilizer states \( |\psi\rangle, |\phi\rangle \in \mathcal{S}_n \). Then \( \langle \psi|\phi\rangle = b^2 p^{-3/2} e^{ixm/4} \) 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 Section IV 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 \( G \subseteq P_n \) is a stabilizer group and \( |\varphi\rangle \in \mathcal{S}_n \). Then \( \Pi_G |\varphi\rangle = b^2 2^{-3/2} |\phi\rangle \) for some \( b = 0, 1 \), some integer \( p \geq 0 \), and stabilizer state \( |\phi\rangle \in \mathcal{S}_n \). One can compute \( b, p, |\phi\rangle \) in time \( O(rn^2) \) as explained in Section IV E. Recall that

\[
r = \dim(G). \quad \text{Finally, one can generate a random stabilizer state drawn from the uniform distribution on } \mathcal{S}_n \text{ in time } O(n^2), \text{ see Section IV D.}
\]

II. CLASSICAL SIMULATION ALGORITHMS

In this Section we describe two classical simulation algorithms. The first algorithm is a classical Monte Carlo algorithm that approximates the probability \( P_{\text{out}}(x) \) for a given string \( x \in \{0, 1\}^w \) with a specified relative error \( \epsilon \) and a failure probability \( p_f \). The algorithm has runtime

\[
\tau = O \left( (w + t)(c + t) + (n + t)^3 + 2^{3t} c^{-2} p_f^{-1} \right), \quad (3)
\]

where \( \beta \leq 1/2 \) is a constant that depends on the implementation details. The second algorithm is the classical simulation algorithm which allows one to sample the output string \( x \) from a distribution which is \( \epsilon \)-close to \( P_{\text{out}} \) with respect to the \( L_1 \)-norm, with runtime given by Eq. (2) in the main text.

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

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

and

\[
\tau_2 = O(2^{3t} c^{-2} p_f^{-1}).
\]

The first stage computes a stabilizer group \( G \subseteq P_t \) and an integer \( u \) such that

\[
P_{\text{out}}(x) = 2^{-u} \langle A^{ot}|\Pi_G|A^{ot}\rangle. \tag{4}
\]

We begin by replacing each \( T \)-gate in the original circuit \( U \) by the well-known gadget [5] shown in Fig. 2 in the main text. 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.

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 $|0^n \otimes A^{\otimes t}\rangle$. Combining the final measurement projector $\Pi(x) = |x\rangle \langle x|_{Q_{\text{out}}} \otimes I_{\text{else}}$ with the projectors $|0\rangle \langle 0|$ acting on the ancillary qubits gives a projector

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

acting on $n + t$ qubits such that

$$P_{\text{out}}(x) = 2^t (|0^n \otimes A^{\otimes t}| V^\dagger IV |0^n \otimes A^{\otimes t}\rangle).$$  \hfill (5)

Here we noted that the postselection probability is $2^{-t}$. Obviously, $\Pi = \Pi(x)$ for a stabilizer group $W \subseteq P_{n+t}$ of dimension $w + t$. Namely, let $q(j)$ be the $j$-th qubit of $Q_{\text{out}}$. Generators of $W$ 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 $V^\dagger IV \Pi V = \Pi_V$, where $V$ is a stabilizer group of dimension $w + t$ generated by $R'_j = V^\dagger R_j V \in P_{n+t}$ 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, $V$ can be computed in time $O((w+t)(c+t))$.

Let $V_0$ be the subgroup of $V$ that includes all Pauli operators which act as $I$ or $Z$ on each of the first $n$ qubits. Let $v = \dim (V_0)$. A generating set $Q_1, \ldots, Q_v \in V_0$ can be computed in time $O(n(w+t) + (w+t)^3) = O((n+t)^3)$ using standard linear algebra. We get

$$\langle 0^n | V^\dagger IV | 0^n \rangle = \langle 0^n | \Pi_V | 0^n \rangle$$

$$= 2^{-w-t+v} \langle 0^n | \Pi_{V_0} | 0^n \rangle.$$  \hfill (6)

since $\langle 0^n | P | 0^n \rangle = 0 \forall P \in V \setminus V_0$. Define $t$-qubit Pauli operators $G_i = \langle 0^n | Q_i | 0^n \rangle$, $i = 1, \ldots, v$. These operators pairwise commute since $V_0$ is abelian and $Q_i$ commute with each other on the first $n$ qubits. If $-I \notin \langle G_1, \ldots, G_v \rangle$ then there exists $Q \in V_0$ with $-I = \langle 0^n | Q | 0^n \rangle$ and therefore

$$\Pi_V | 0^n \rangle = -\Pi_V | 0^n \rangle = 0$$

in which case $P_{\text{out}}(x) = 0$ and we are done. Let us now consider the case $-I \notin \langle G_1, \ldots, G_v \rangle$. In this case let $G \subseteq P_t$ be the stabilizer group generated by $G_1, \ldots, G_v$ and $r = \dim (G)$. One can check the condition $-I \notin \langle G_1, \ldots, G_v \rangle$ and compute in time $O(t^3)$. Without loss of generality, $G = \langle G_1, \ldots, G_r \rangle$. Noting that $V_0$ must contain $2^{w-r}$ elements acting trivially on the last $t$ qubits yields $\langle 0^n | \Pi_{V_0} | 0^n \rangle = 1/2^r$. This proves Eq. (4) with $u = w - v$ and the stabilizer group $G$ defined above. Combining all the steps needed to compute $G$ gives the promised runtime $t_1 = O((w + t)(c + t) + (n + t)^3)$.

The second stage of the algorithm computes the expectation value in Eq. (4) by decomposing $|A^{\otimes t}\rangle$ into a linear combination of stabilizer states. Suppose

$$|A^{\otimes t}\rangle = \sum_{a=1}^{\chi} y_a |\varphi_a\rangle$$  \hfill (7)

for some stabilizer states $|\varphi_a\rangle \in \mathcal{S}_t$ 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 $|\phi_a\rangle \in \mathcal{S}_t$ such that

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

see Section IV E for details. As stated above, this computation takes time $O(\chi^3)$. Introducing new coefficients $z_a = 2^{-w+p_a/2}$ and using Eqs. (4, 7) one gets

$$P_{\text{out}}(x) = ||\psi||^2, \quad |\psi\rangle = \sum_{a=1}^{\chi} z_a |\phi_a\rangle, \quad \phi_a \in \mathcal{S}_t.$$  \hfill (8)

Here $|\phi_a\rangle$ are $t$-qubit stabilizer states. Using the norm estimation subroutine described in the main text of the paper, $P_{\text{out}}(x) = ||\psi||^2$ can be approximated in time $O(\chi^3 c^2 \epsilon^{-2} p_f^{-1})$, as promised.

Since the runtime grows linearly with $\chi$, we would like to choose a stabilizer decomposition in Eq. (7) with a small rank $\chi$. 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 $\chi_t$ is unknown. Using the identity

$$|A^{\otimes 2}\rangle = \frac{1}{2} (|00\rangle + i|11\rangle) + \frac{e^{ix/4}}{2} (|01\rangle + |10\rangle)$$  \hfill (9)

one can see that $|A^{\otimes 2}\rangle$ 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. (9) to each pair one gets $\chi_t \leq 2^t/2$. The results of [4] give a slightly better bound $\chi_t \leq 2^t$ with $\beta \approx 0.47$. This completes the analysis of the first algorithm.

Remark 1: One may attempt to derive a stronger bound on the failure probability $p_f$ in the norm estimation algorithm. Namely, let $\xi_H$ be a random variable defined in Eq. (7) in the main text where the states $|\theta_t\rangle$ are drawn from the Haar measure. Clearly, $\xi$ and $\xi_H$ have the same mean, $\xi_t = \xi = ||\psi||^2$. Let $p_f = Pr[|\xi - \xi_H| > \epsilon \xi]$ . Then $p_f \leq 2\exp(-C\epsilon^2)$, where $C = O(1)$ is a constant, see Lemma II.3 of Ref. [6].

If a similar tail bound applies to the random variable $\xi$, one could achieve a failure probability $p_f$ in the norm estimation algorithm using only $L = O(\epsilon^{-2} \log (p_f^{-1}))$ samples.

Remark 2: If $G$ has a small dimension, namely, $r < \beta$, it can be easier to compute $P_{\text{out}}(x)$ directly from Eqs. (1, 4) which yield $P_{\text{out}}(x) \sim \sum_{P \in G} \langle A^{\otimes t}| P |A^{\otimes t}\rangle$. Clearly, each term in the sum can be computed in time $O(t)$, so the overall runtime becomes $O(!\mathcal{G}) = O(t^{2^2})$.

Remark 3: An alternative strategy to estimate the expectation value in Eq. (4) is to compute the inner products

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

for $i = 1, \ldots, L$. Here $|\varphi_a\rangle$ are the stabilizer states defined in Eq. (7) and $|\theta_i\rangle$ are random stabilizer states. The
same arguments as above show that
\[ \langle A^\otimes t | \Pi_G | A^\otimes t \rangle = \| \Pi_G | A^\otimes t \| ^2 \approx \frac{d}{L} \sum_{i=1}^{L} \| \Pi_G \theta_i | A^\otimes t \| ^2 . \]
This may be beneficial in the regime \( L \ll \chi \) since one has to compute the action of \( \Pi_G \) only \( L \) times rather than \( \chi \) times.

Let us now describe the algorithm that allows one to sample \( x \) from the distribution \( P_{\text{out}} \) with statistical error \( \epsilon \). As before, we replace each \( T \)-gate in the original circuit \( U \) by the gadget shown on Fig. 2 in the main text, prepare all magic states \( |A\rangle \) at the very first time step, and permute the qubits such that the initial state is \( |0^\otimes n \otimes A^\otimes t \rangle \). 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 \( \epsilon + t + y \) 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 \( \Phi \) is applied to a state \( \rho_{in} = |0\rangle\langle 0| \otimes |A\rangle\langle A| \otimes I^\otimes t \). Then the final state of the \( n \) computational qubits is \( U|0^\otimes n \rangle \) regardless of \( y \) and each \( y \) appears with probability \( 2^{-t} \). Thus
\[ \Phi(\rho_{in}) = U|0^\otimes n \rangle \otimes |A\rangle^\otimes t \otimes I^\otimes 2 . \tag{10} \]
Next suppose that \( \Phi \) is applied to a state \( \rho_{in} = |0\rangle\langle 0| \otimes |\psi\rangle\langle \psi| \). where \( |\psi\rangle \) is a linear combination of \( \chi \) stabilizer states \( |\varphi_1\rangle, \ldots, |\varphi_\chi\rangle \in S_t \) that approximates \( |A^\otimes t \rangle \) with a small error:
\[ |\psi\rangle = \sum_{a=1}^{\chi} z_a |\varphi_a\rangle , \quad \| A^\otimes t |\psi\rangle \| ^2 \geq 1 - \epsilon^2 / 25 . \tag{11} \]
Here \( z_a \) are some coefficients and we assume \( |\psi\rangle \) has unit norm. The error \( \epsilon^2 / 25 \) is sufficient to ensure that the output distribution of the overall simulation algorithm is \( \epsilon \)-close to \( P_{\text{out}} \). From Eq. (11) one gets
\[ \| \rho_{in} - \tilde{\rho}_{in} \| _1 \leq \| |A\rangle\langle A| \otimes t - |\psi\rangle\langle \psi| \| _1 \leq \frac{2}{5} \epsilon . \tag{12} \]
By definition of \( \Phi \),
\[ \Phi(\tilde{\rho}_{in}) = \sum_y p_y |\phi_y\rangle\langle \phi_y| \otimes |y\rangle\langle y| , \tag{13} \]
where
\[ p_y = \langle 0^\otimes n \otimes \psi| V_y^\dagger (I_n \otimes |y\rangle\langle y|) V_y |0^\otimes n \otimes \psi \rangle \tag{14} \]
and \( |\phi_y\rangle \) are normalized \( t \)-qubit states defined by
\[ |\phi_y\rangle = \rho_y^{-1/2} \langle y| V_y |0^\otimes n \otimes \psi \rangle . \tag{15} \]
Clearly, \( p \) is a normalized probability distribution on the set of \( t \)-bit strings. The state \( |\phi_y\rangle \) is defined only for \( p_y > 0 \). Combining Eqs. (10,12,13) and tracing out the last \( t \) qubits of \( \Phi(\rho_{in}) \) and \( \Phi(\tilde{\rho}_{in}) \) one infers that
\[ \| U|0^\otimes n \rangle \otimes |A\rangle^\otimes t - \sum_y p_y |\phi_y\rangle\langle \phi_y| \| _1 \leq \frac{2}{5} \epsilon . \tag{16} \]
Here we noted that TPCP maps do not increase the trace distance. Combining Eqs. (10,12,13) and tracing out the first \( n \) qubits of \( \Phi(\rho_{in}) \) and \( \Phi(\tilde{\rho}_{in}) \) shows that the distribution \( p \) satisfies \( \| p - u \| _1 \leq \frac{\epsilon}{5} \), where \( u \) is the uniform distribution on the set of \( t \)-bit strings. Using this fact and Eq. (16) we arrive at
\[ \| U|0^\otimes n \rangle \otimes |A\rangle^\otimes t - \frac{1}{2^t} \sum_y |\phi_y\rangle\langle \phi_y| \| _1 \leq \frac{4}{5} \epsilon . \tag{17} \]
For each \( t \)-bit string \( y \) define a probability distribution \( P_{\text{out}}^y(x) = \langle \phi_y| \Pi(x) |\phi_y\rangle \). Below we give an algorithm which takes as input \( y \) and \( \epsilon \) and produces a sample from a distribution \( \tilde{P}_{\text{out}}^y \) which satisfies
\[ \| P_{\text{out}}^y(x) - \tilde{P}_{\text{out}}^y(x) \| _1 \leq \epsilon / 5 \tag{18} \]
Our algorithm to approximately sample from \( P_{\text{out}} \) has two steps. We first generate a random uniformly distributed \( t \)-bit string \( y \) and then we sample \( x \) from \( \tilde{P}_{\text{out}}^y \). From Eqs. (17,18) we see that the distribution over outputs \( x \in \{0, 1\}^w \) produced by this algorithm approximates \( P_{\text{out}} \) within error \( \epsilon \) in the trace norm.

We are now ready to describe how to sample from \( \tilde{P}_{\text{out}}^y \) satisfying Eq. (18). We first describe how to compute an approximation to \( P_{\text{out}}^y(x) \) with relative error \( \delta \). Note that
\[ P_{\text{out}}^y(x) = \frac{\langle 0^\otimes n \otimes \psi| V_y^\dagger (I_n \otimes |y\rangle\langle y|) V_y |0^\otimes n \otimes \psi \rangle}{\langle 0^\otimes n \otimes \psi| V_y^\dagger (I_n \otimes |y\rangle\langle y|) V_y |0^\otimes n \otimes \psi \rangle} . \tag{19} \]
Here we used Eqs. (14,15). Repeating the same arguments as in the derivation of Eq. (4) one gets
\[ P_{\text{out}}^y(x) = \frac{2^{-w} \langle \psi| \Pi_G |\psi\rangle}{2^{-w} \langle \psi| \Pi_H |\psi\rangle} \tag{20} \]
for some stabilizer groups \( G \subseteq P_t \) 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 \( \langle \psi| \Pi_G |\psi\rangle \) and \( \langle \psi| \Pi_H |\psi\rangle \) with a relative error \( \delta \) in time \( \tau_2 = O(\chi^3 \delta^{-2} p_f^{-1}) \). Recall that \( p_f \) is the probability that the algorithm does not achieve the desired approximation. Thus we can compute \( P_{\text{out}}^y(x) \) with a relative error \( 2\delta \) in time \( \tau_1 + \tau_2 \).

Now consider the task of sampling from \( \tilde{P}_{\text{out}}^y \). Assume for simplicity that \( Q_{out} = \{1, 2, \ldots, w\} \). For each \( j = 1, \ldots, w - 1 \) define conditional probabilities
\[ P_{\text{out}}^y(z|x_1, \ldots, x_{j-1}, z) = \frac{P_{\text{out}}^y(x_1, \ldots, x_{j-1}, z)}{P_{\text{out}}^y(x_1, \ldots, x_{j-1})} , \tag{21} \]
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 $P_{\text{out}}^j(0|x_1, \ldots, x_{j-1})$. 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 $P_{\text{out}}^j(0|x_1, \ldots, x_{j-1})$ 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 $P_{\text{out}}^j(0|x_1, \ldots, x_{j-1})$ and $P_{\text{out}}^j(1|x_1, \ldots, x_{j-1})$ respectively with relative error $O(\delta)$. Such an approximation $q_j$ can be computed in time $O(t_1 + t_2)$ using the procedure described above for approximating the probabilities on the right-hand side of Eq. (21).

We now analyze the resulting simulation algorithm and show that we can ensure that $P_{\text{out}}^j$ in total and choosing $q_j = a$ if $b < a$ and we set $q_j = 1 - b$.

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

The overall running time of this algorithm is $t_1^* + t_2^*$, where $t_1^* = O(w T_1) = O(w(w + t)(c + t) + w(n + t)^3)$ and $t_2^* = O(w T_2) = O(\chi w^4 t^3 \epsilon^{-3})$.

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 in the main text, 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 [7](otherwise it requires seven $T$-gates [8, 9]). 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.

III. APPROXIMATING MAGIC STATES

In this section we show how to compute a decomposition Eq. (11) with $\chi = O(2^{\gamma} \epsilon^{-2})$, where $\gamma$ satisfies Eq. (3) in the main text. 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, $|A\rangle = e^{i\pi/8}HS|H\rangle$. Thus it suffices to construct a state $|\psi\rangle = \sum_{a=1}^{\chi} z_a |\varphi_a\rangle$, $\varphi_1, \ldots, \varphi_{\chi} \in S_t$ (22) such that $\|\langle \psi |\| = 1$.

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

for all sufficiently small $\delta > 0$.

Our starting point is the identity $|H^{\otimes t}\rangle = \frac{1}{\sqrt{2^{t}}} \sum_{x \in F_2^t} |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \ldots \otimes \tilde{x}_t\rangle$ (24) where $|0\rangle \equiv |0\rangle$, $|\tilde{1}\rangle \equiv |H\rangle|0\rangle = 2^{-1/2}(|0\rangle + |1\rangle)$, and $\nu \equiv \cos(\pi/8)$.

The right-hand side of Eq. (24) is a uniform superposition of $2^t$ non-orthogonal stabilizer states labeled by elements of the vector space $F_2^t$. We shall construct an approximation $|\psi\rangle$ 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 $F_2^t$.

Let $L(t, k)$ be the set of all $k$-dimensional linear subspaces $L \subseteq F_2^t$. We will fix $k$ below. For each $L \in L(t, k)$ define a state $|\mathcal{L}\rangle = \frac{1}{Z(\mathcal{L})} \sum_{x \in \mathcal{L}} |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \ldots \otimes \tilde{x}_t\rangle$ (25)

where

\[ Z(\mathcal{L}) \equiv \sum_{x \in \mathcal{L}} 2^{-|x|/2}. \]

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, $|\mathcal{L}\rangle|\mathcal{L}\rangle = 1$. We take our approximation $|\psi\rangle$ from Eq. (22) to be Eq. (25) for a suitably chosen subspace $L^* \in L(t, k)$, which gives an approximate decomposition of $|H^{\otimes t}\rangle$ using $\chi = 2^k$ stabilizer states. How small can we hope to make $k$?

Using the fact that $\langle H|0\rangle = \langle H|1\rangle = \nu$ we see that

\[ |\langle H^{\otimes t}|\mathcal{L}\rangle|^2 = \frac{2^{2t} \nu^{2t}}{Z(\mathcal{L})} \]

(27)
From this we immediately get a lower bound on $k$. Indeed, since $Z(L) \geq 1$ 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 $L^*$ with $2^k = O(\delta^{-1} \nu^{-2t})$. Thus for constant $\delta$ we achieve the best possible scaling of $k$ with $t$. We will use the following fact about random subspaces of $F_2^n$.

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

$$E(Z(L)) \leq 1 + 2^k \nu^{2t}. \quad (28)$$

**Proof.** By linearity, we have

$$E(Z(L)) = 1 + \sum_{x \in F_2^n \setminus \emptyset} 2^{-|x|/2} \cdot E(\chi_L(x)), \quad (29)$$

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

$$E(Z(L)) = 1 + \frac{(2^k - 1)}{(2^t - 1)} \sum_{x \in F_2^n \setminus \emptyset} 2^{-|x|/2}$$

$$= 1 + \frac{(2^k - 1)}{(2^t - 1)} (2^t \nu^{2t} - 1)$$

$$\leq 1 + 2^k \nu^{2t}.$$ 

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

$$4 \geq 2^k \nu^{2t} \delta \geq 2. \quad (30)$$

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

$$\Pr \left[ \frac{Z(L)}{(1 + 2^k \nu^{2t})(1 + \delta/2)} \geq 1 \right] \leq \frac{E(Z(L))}{(1 + 2^k \nu^{2t})(1 + \delta/2)} $$

$$\leq 1 - \frac{\delta}{2 + \delta}. \quad (31)$$

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

$$Z(L^*) \leq (1 + 2^k \nu^{2t})(1 + \delta/2) \quad (31)$$

with constant probability. Plugging Eq. (31) into Eq. (27) we see that

$$\langle H^{\otimes t} | L^* \rangle^2 \geq \frac{1}{1 + 2^{-k} \nu^{-2t}} \quad (32)$$

$$\geq \frac{1}{1 + \delta/2} \quad (33)$$

$$\geq 1 - \delta,$$

where in the second line we used Eq. (30). The state $|\psi\rangle = |L^*\rangle$ obtained in this way therefore satisfies Eq. (23) with

$$\chi = 2^k \leq 4 \nu^{-2t} \delta^{-1} = O(\nu^{-2t} \delta^{-1}) = O(2^t \nu \delta^{-1}). \quad (32)$$

This algorithm has running time $O(t \nu^{-2t} \delta^{-2})$, since we must check the condition Eq. (31) for each of the $O(\delta^{-1})$ randomly sampled elements of $L(t,k)$ (note that the time required to sample each element is $O(\text{poly}(t))$).

**Remark:** One may ask whether a stronger bound on $\chi$ can be obtained by truncating the expansion of $|H^{\otimes t}\rangle$ in some other basis of stabilizer states. For example, consider the standard 0,1-basis of $t$ qubits. The expansion of $|H^{\otimes t}\rangle$ in this basis is concentrated on basis vectors $x \in F_2^n$ with Hamming weight $|x| = (1 - \nu^2) t + O(t^1/2)$. The number of such basis vectors scales as $\chi \sim 2^t H_2(\nu) \approx 2^t \nu^t$, which is the binary Shannon entropy function. Thus replacing the 0,1-basis by the 0,1-basis gives a significantly worse bound on $\chi$.

As noted above, taking $\delta$ to be a constant our construction has the best possible scaling $\chi = O(\nu^{-2t})$ of any decomposition of the form Eq. (25). In fact, we prove the following lower bound on the stabilizer rank of $|H^{\otimes t}\rangle$.

**Lemma 2.** Consider a state $|\psi\rangle = \sum_{a \in \mathbb{F}_2} z_a |a\rangle$, where $|\phi_a\rangle \in S_t$. Suppose $\|\psi\| = 1$ and $\|\langle \psi | H^{\otimes t}\rangle\| \geq f$. Then $\chi \geq \nu^{-2t} f^2 \|z\|^{-2}$, where $z = (z_1, \ldots, z_t) \in \mathbb{C}^t$.

**Proof.** First, let us show that

$$F_1 = \max_{\phi \in S_t} \|\langle \phi | H^{\otimes t}\rangle\| = \nu^t. \quad (33)$$

The lower bound $F_1 \geq \nu^t$ 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 $|\phi\rangle \in S_t$. Suppose we measure the first qubit of $|\phi\rangle$ 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 $|\phi\rangle$. Consider three cases.

**Case 1:** $P_0 = 1$. Then $|\phi\rangle = |0\rangle \otimes \psi$ for some $|\psi\rangle \in S_{t-1}$ and $F_t = \nu \|\langle \psi | H^{\otimes (t-1)}\rangle\| \leq \nu F_{t-1}$.

**Case 2:** $P_0 = 0$. Then $|\phi\rangle = |1\rangle \otimes \psi$ for some $|\psi\rangle \in S_{t-1}$ 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 $|\phi\rangle = 2^{-1/2} (|0\rangle \otimes \psi_0 + |1\rangle \otimes \psi_1)$ for some $|\psi_0\rangle, |\psi_1\rangle \in S_{t-1}$. 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. (33). From Eq. (33) one gets

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

This is equivalent to the statement of the lemma.

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

A. Quadratic forms

The remaining sections provide more details on implementation of our algorithms. Section IV 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 Section IV B. Then we present algorithms for computing the inner product between stabilizer states is established by the following lemma. Section IV A presents some basic facts about quadratic forms over finite fields and computations of our algorithms. Section IV A presents some basic facts about quadratic forms over finite fields and computations of our algorithms. Section IV A presents some basic facts about quadratic forms over finite fields and computations of our algorithms.

The function $q(x, y)$ defined by Eq. (34) is a symmetric quadratic form that takes values 0, 4 (mod 8).

Namely, $J(x, y) = J(x, y) = J(0, 0) = 0, J(x', x'', y) = J(x', y) + J(x'', y)$, and $J(x, y) = 0, 4$ (mod 8) for all $x, x', x'', y, y \in L(K)$.

Proof. Let $x = x' + x''$. Substituting $x \leftarrow x'$ in Eq. (34) gives

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

Substituting $x \leftarrow x''$ and $z \leftarrow z + x'$ in Eq. (34) gives

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

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. (34). Replacing $z$ by $z + x$ in Eq. (34) yields

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

Combining this and Eq. (34) one gets $2J(x, y) = 0$, that is, $J(x, y) = 0$ (mod 4).

As a corollary, one gets $q(x + y) - q(z) \in \{0, 2, 4, 6\}$ for all $z \in K$ and for all $x \in L(K)$. This can be checked by choosing $x = y$ in Eq. (34) and using the fact that $J(x, x) \in \{0, 4\}$.

Suppose $g_1, \ldots, g_k \in L(K)$ is some fixed basis of $L(K)$, $h \in K$ is some fixed shift vector, and $x \in K$. Then

$$x = h \oplus x_1 g_1 \oplus \ldots \oplus x_k g_k, \quad x_i \in \{0, 1\}.$$
Proof. The claim that any stabilizer state can be written in the form Eq. (39) follows from the explicit characterization of quadratic forms Eqs. (35,36,37) and the canonical form of stabilizer states derived in Refs. [13–15]. The uniqueness of the decomposition Eq. (39) 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

$$g^a \leftarrow \sum_{b=1}^{k} R_{a,b} g^b \pmod{2}, \quad (40)$$

where $1 \leq a \leq k$. The shift vector $h$ remains unchanged. Applying Eq. (35) 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 $Q \leftarrow Q,$

$$D_a \leftarrow \sum_{b=1}^{k} R_{a,b} D_b + \sum_{1 \leq b < c \leq k} J_{b,c} R_{a,b} R_{a,c}, \quad (41)$$

and

$$J \leftarrow RJ R^T. \quad (42)$$

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

$$h \leftarrow h \oplus y, \quad \text{where} \quad y = \sum_{a=1}^{k} y_a g^a \pmod{2}. \quad (43)$$

Using Eq. (36) one can easily check that the coefficients $(Q, D, J)$ transform according to

$$Q \leftarrow Q + \sum_{a=1}^{k} D_a y_a + \sum_{1 \leq a < b \leq k} J_{a,b} y_a y_b, \quad (44)$$

$$D_a \leftarrow D_a + \sum_{b=1}^{k} J_{a,b} y_b, \quad (45)$$

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. (40) 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. (41,42) 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 the overall time is $O(|R|^2)$. We conclude that computing the representation $(Q, D, J)$ of $q$ in the new basis takes time

$$\tau_{update} = O(\min (k^3, |R|^2)). \quad (46)$$

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

$$W(q) \equiv \sum_{x \in \mathbb{F}_2^k} e^{i\pi q(\vec{x})}, \quad (47)$$

where $q(\vec{x})$ is defined by Eq. (35). Of course, the addition in Eq. (47) 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 Section IV C.

It will be convenient to consider a more general sum

$$W(\mathcal{K}, q) = \sum_{x \in \mathcal{K}} e^{i\pi q(\vec{x})}, \quad (48)$$

where $\mathcal{K} = \mathcal{L}(\mathcal{K}) \oplus h$ is an affine space and $q : \mathcal{K} \rightarrow \mathbb{Z}_8$ is a quadratic form on $\mathcal{K}$. Clearly, Eq. (47) is a special case of Eq. (48). 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

$$[k] = D_1 \cup \ldots \cup D_r \cup M \cup S, \quad (49)$$

such that

$$|D_1| = \ldots = |D_r| = 2, \quad |S| \leq 1, \quad (50)$$

$$J_{a,a} = \begin{cases} 0 & \text{if } a \notin S, \\ 4 & \text{if } a \in S. \end{cases} \quad (51)$$

$$a \in M \quad \Rightarrow \quad J_{a,b} = 0 \quad \forall b \in [k] \setminus S, \quad (52)$$

and $D_i = \{a, b\}$ implies

$$J_{a,b} = 4 \quad \text{and} \quad J_{a,c} = J_{b,c} = 0 \quad \forall c \notin S \cup \{a, b\}. \quad (53)$$

Some of the subsets in Eq. (49) can be empty. Let us write

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

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. (35) and using Eqs. (50,52,53) one can check that

$$q(\vec{x}) = Q + \sum_{j=1}^{r} q_j(x_{a(j)}, x_{b(j)}) + \sum_{c \in M} D_c x_c. \quad (54)$$

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

$$q_j(y,z) = 4yz + D_{a(j)} y + D_{b(j)} z. \quad (55)$$
Examination of Eqs. (48,54,55) 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,

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

where

\[
\Gamma_j = 1 + e^{i \frac{\pi}{4} D_{a(j)}} + e^{i \frac{\pi}{4} D_{b(j)}} = e^{i \frac{\pi}{4} (D_{a(j)} + D_{b(j)})}
\]

Combining Eqs. (56,57) one can compute \(W(\mathcal{K}, q)\) in time \(O(k)\).

Consider now the remaining case \(S \neq \emptyset\). Since \(|S| \leq 1\), we have \(S = \{s\}\) for some \(s \in [k]\). By repeatedly applying Eq. (35) and using Eqs. (50,52,53) one can check that

\[
q(x) = Q + D_s x + \sum_{j=1}^r q_j(x_{a(j)}, x_{b(j)}, x_s)
\]

\[
+ \sum_{c \in M} (D_c x_c + J_{c,s} x_c x_s),
\]

where \(q_j : \mathbb{F}_2^3 \rightarrow \mathbb{Z}_8\) is defined by

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

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

\[
W_\sigma = \sum_{x \in \mathbb{Z}^k : x_s = \sigma} e^{i \frac{\pi}{4} q(x)}, \quad \sigma = 0, 1.
\]

Examination of Eqs. (58,59) reveals that \(W_\sigma\) factorizes into a product of \(O(k)\) terms such that each term can be computed in time \(O(1)\). Specifically,

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

where

\[
\Gamma_j(\sigma) = 1 + \exp \left[ i \frac{\pi}{4} (J_{a(j),s} \sigma + D_{a(j)}) \right]
\]

\[
+ \exp \left[ i \frac{\pi}{4} (J_{b(j),s} \sigma + D_{b(j)}) \right]
\]

\[- \exp \left[ i \frac{\pi}{4} (J_{a(j),s} \sigma + J_{b(j),s} \sigma + D_{a(j)} + D_{b(j)}) \right].
\]

Combining Eqs. (61,62) one can compute \(W_0 + W_1\) in time \(O(k)\).

To transform an arbitrary basis \(g^1, \ldots, g^k\) of \(\mathcal{L}(K)\) into the canonical form we shall use a version of the Gram-Schmidt orthogonalization. It involves at most \(k\) basis changes Eq. (40) 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. (46).

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 \(a \in S \setminus \{s\}\). From Eq. (41) one gets \(D_a \leftarrow D_a + D_s + J_{a,s} \in \{0, 4, 6\}\) for all \(a \in S \setminus \{s\}\) and \(D_a \leftarrow D_a\) for all \(a \notin S\). Set \(S = \{s\}\). Now we can assume that \(D_a \in \{0, 4, 6\}\) for all \(a \notin S\) for some subset \(S\) such that \(|S| \leq 1\). From Eq. (38) we infer

\[
J_{a,a} = 0 \quad \text{for all } a \notin S.
\]

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

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

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 \(J\) corresponding to \(J\) such that \(J_{a,b} = 1\) if \(J_{a,b} = 4\) and \(J_{a,b} = 0\) otherwise. Perform a basis change

\[
g^c \leftarrow g^c \oplus J_{a,c} g^b \oplus J_{b,c} g^a \quad \text{for all } c \in E \setminus \{a, b\}.
\]

Using Eq. (63) one can check that the new basis vectors obey \(J(g^c, g^d) = J(g^c, g^e) = 0\) 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. (49). By repeating the above steps at most \(k\) times one makes \(E = \emptyset\). Furthermore, the \(R\) matrices corresponding to the basis change Eq. (65) 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. (40) with sparse matrices \(R\) such that \(|R| = O(k)\). This has cost \(O(k|R|^2) = O(k^3)\). We summarize the algorithm below.
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. (41,42). 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. [4], 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 $W(\mathcal{K}, q) = \epsilon \cdot 2^{p/2} \cdot e^{\pi m/4}$. 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.

B. Standard form of stabilizer states

Suppose $|K, q\rangle \in S_n$ is a stabilizer state of $n$ qubits defined in Eq. (39). 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 
\[
(n, k, h, G, \bar{G} \in \mathbb{F}_{2^n}^{n \times n}),
\]
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,
\[
G\bar{G}^T = I \pmod{2}.
\]
We shall write $g^a$ and $\bar{g}^a$ for the $a$-th row of $G$ and $\bar{G}$ respectively. Thus $\mathcal{L}(\mathcal{K}) = \text{span}(g^1, \ldots, g^k)$ and $(g^a, \bar{g}^a) = \delta_{a,b}$ for $1 \leq a, b \leq n$. 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. (35,36,37), with the shift vector $h$. Thus, a stabilizer state $|K, q\rangle$ 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. (66) and Eq. (38).

We shall often use a subroutine that alters a stabilizer state $|K, q\rangle$ 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
\[
\mathcal{M} = \mathcal{K} \cap \{x \in \mathbb{F}_2^n : (\xi, x) = \alpha\}. \tag{67}
\]
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 $|K, q\rangle$ 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
\[
\bar{g}^i \leftarrow \bar{g}^i \oplus \sum_{a \in S} \bar{g}^a.
\]
Thus vectors coefficients \( O \) where ing Eqs. (41,42) takes time \( O(k^2) = O(kn) \). Now basis vectors \( g^1, \ldots, g^k \) are orthogonal to \( \xi \) and \( (\xi, g^k) = 1 \). Thus
\[
\mathcal{M} = h' \oplus \text{span}(g^1, \ldots, g^{k-1}) \equiv h' \oplus \mathcal{L}(\mathcal{M}),
\]
where \( h' = h \oplus \beta g^k \) is the new shift vector. Update the coefficients \( Q, D, J \) using Eqs. (41,42) takes time \( O(k^2) = O(kn) \). 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(|K, q⟩, ξ, α)
    S ← \{a ∈ [k] : (ξ, g^a) = 1\}
    β ← α ⊕ (ξ, h)
    if S = ∅ and β = 1 then
        return EMPTY
    end if
    if S = ∅ and β = 0 then
        return SAME
    end if
    Pick any \( i \in S \)
    S ← S \ \{i\}
    for \( a ∈ S \) do
        \( g^i ← g^i \oplus g^a \)
        Update \((D, J)\) using Eqs. (41,42)
    end for
    \( g^i ← g^i \oplus \sum_{a ∈ S} g^a \)
    Swap \( g^i \) and \( g^k \).
    Update \((D, J)\) using Eqs. (41,42)
    \( h ← h \oplus \beta g^k \)
    Update \((Q, D)\) using Eqs. (44,45)
    Remove the \( k \)-th row/column from \( J \)
    Remove the \( k \)-th element from \( D \)
    \( k ← k - 1 \)
    return SUCCESS
end function
```

To simplify notations, here we assume that the function \( \text{SHRINK} \) modifies the data describing the initial state. The function reports whether the new affine space \( \mathcal{K} \) is empty or the same as the initial space. It reports SUCCESS 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 \( \text{SHRINK}' \) for such lazy version.

**C. The inner product**

Consider a pair of \( n \)-qubit stabilizer states
\[
|φ_α⟩ = |K_α, g_α⟩, \quad α = 1, 2
\]
with the standard forms \((n, k_α, h_α, G_α, \bar{G}_α, Q_α, D_α, J_α)\).

Below we describe an algorithm that computes the inner product
\[
⟨φ_2|φ_1⟩ = 2^{-(k_1+k_2)/2} \sum_{x \in K_2 \cap J_2} e^{i\pi f(q_2(x))}. \quad (68)
\]
in time \( O(n^3) \). First we note that \( x ∈ K_2 \) iff
\[
x ⊕ h_2 ∈ \mathcal{L}(K_2) = \text{span}(g^1, \ldots, g^k).
\]
Thus \( x ∈ K_2 \) iff \( x ⊕ h_2 \) is orthogonal to all dual basis vectors \( g^b_2 \) with \( k_2 < a ≤ n \). Here and below \( g^a_2 \) and \( g^b_2 \) denote the \( b \)-th row of \( G_α \) and \( G_α \) respectively. Thus
\[
\mathcal{K} ≡ K_1 \cap K_2 = \bigcap_{b=k_2+1}^{n} \{ x ∈ K_1 : (\bar{g}_2^b, x) = (h_2, g_2^b) \}.
\]

One can compute the standard form of \(|K_1, q_1⟩\) by \( n - k_2 \) calls to the function \( \text{SHRINK} \) defined in Section IV B with \( ξ = g^2_2 \) and \( α = (h_2, g^2_2) \) for \( b = k_2 + 1, \ldots, n \). This takes time
\[
τ_1 = O((n - k_2)k_1 n)
\]
since we have to call \( \text{SHRINK} \) \( n - k_2 \) times.

Let \( \mathcal{K} = (n, k, h, G, G) \) 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
\[
h = h_2 + \sum_{a=1}^{k_2} y_α g^a_2, \quad \text{where} \ y_α = (h ∧ h_2, g^a_2).
\]

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. (44,45). This takes time \( O(k_2^2) \). A simple algebra shows that \( \mathcal{L}(\mathcal{K}) = \mathcal{L}(K_2) ∩ \mathcal{L}(K_2) \), that is, \( g^a ∈ \mathcal{L}(K_2) \) for all \( a = 1, \ldots, k \). Define a matrix \( R \) of size \( k \times k_2 \) such that
\[
g^a = \sum_{b=1}^{k_2} R_{a,b} g^b_2 \pmod{2}, \quad 1 ≤ a ≤ k.
\]

Using the dual basis of \( K_2 \) one gets \( R_{a,b} = (g^a_2, g^b_2) \). 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. (41,42) which takes time \( O( k k_2 ) \), see Eq. (46). (Here we used a slightly stronger version of Eq. (46) taking into account that \( R \) is a rectangular matrix.) The runtime up to this point is
\[
τ_2 = τ_1 + O(kk_2 n).
\]

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 \( q ≡ q_1 − q_2 \) has coefficients \((Q, D, J)\), where \( Q = Q_1 − Q_2, D = D_1 − D_2, \) and \( J = J_1 − J_2 \). We get
\[
⟨φ_2|φ_1⟩ = 2^{-(k_1+k_2)/2}. W(Q, D, J),
\]
where $W(Q,D,J)$ is the exponential sum Eq. (47) that can be computed in time $O(k^3)$, see Section IV A. The overall running time is thus

$$
\tau = \tau_2 + O(k^3) = O((n-k_2)k_1n + kk_2n + k^3) = O(n^3).
$$

We summarize the entire inner product algorithm below.

```
function InnerProduct((|K_1,q_1>, |K_2,q_2>))
    K ← K_1
    for b = k_2 + 1 to n do
        α ← (b_2, g_2^b)
        ε ← SHRINK((|K,q>, g_2^b, α)
        if ε = EMPTY then
            return 0
        end if
    end for
    ▷ Now K = K_1 ∩ K_2 = (n, k, h, G, G)
    for a = 1 to k_2 do
        y_a ← (h ⊕ h_2, g_2^a)
        for b = 1 to k do
            R_{0,a} ← (y^b, g_2^b)
        end for
        h_2 ← h_2 ⊕ \sum_{b=1}^{k_2} y_a g_2^b = h
        Update (Q_2, D_2) using Eqs. (44,45) with y
        Update (D_2, J_2) using Eqs. (41,42) with R
        ▷ Now q_1, q_2 are defined in the same basis
        Q ← Q_1 − Q_2
        D ← D_1 − D_2
        J ← 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 $(\epsilon, p, m)$ such that $|\phi_2| |\phi_1> = \epsilon \cdot 2^p / e^{\pi m / 4}$. If both $k_1$ and $k_2$ are small, one can compute the intersection $K_1 \cap K_2$ 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 $K$ 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

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

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 $k_2 \approx n$, see Section IV 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 $|\tilde{x}\rangle |\phi>$, where $\phi \in S_n$ is drawn from the uniform distribution (as described in Section IV D), $x \in F_2^n$ is a random uniformly distributed string, and $|\tilde{x}\rangle \equiv |\tilde{x}_1 \otimes \tilde{x}_2 \otimes \cdots \otimes \tilde{x}_n>$, where $|0\rangle = |0\rangle$ and $|1\rangle = H|0\rangle$. This choice is justified since our simulation algorithm only requires inner products of the above form.

### D. Random stabilizer states

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

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

$$
S^k_n = \{ |K,q> \in S_n : \dim (K) = k \}.
$$

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

$$
P(d) = \frac{\left| S^{n-d}_n \right|}{\sum_{m=0}^{n} \left| S^m_n \right|}
$$

and generates a random subspace $K \subseteq F_2^n$ of dimension $k = n - d$. To compute $P(d)$ we need the following fact.

**Lemma 5.**

$$
\left| S^{n-d}_n \right| = 8 \cdot 2^{n+1 \cdot |n/(n+1)-d/(d+1)|} \prod_{a=1}^{d} \frac{1 - q^{d-n-a}}{1 - 2^{-a}}.
$$

for any $d = 1, \ldots, n$ and $\left| S^0_n \right| = 8 \cdot 2^{n+1 \cdot |n/(n+1)|}$.

**Proof.** Let $k \equiv n-d$. The number of $k$-dimensional linear subspaces $L \subseteq F_2^n$ is known to be

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

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

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

quadratic forms $q : K \rightarrow \mathbb{Z}_2$. Here the three factors represent the number of choices for the coefficients $(Q,D,J)$ in Eqs. (35,36,37) respectively (recall that the diagonal of $J$ is determined by $D$, see Eq. (38)). It follows that $\left| S^k_n \right| = 2^{n-k} \cdot \Gamma_n^k \cdot \Lambda_n^k$, which gives Eq. (71). \hfill \Box

One can rewrite Eq. (70) as

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

where $\eta(0) = 1$ and

$$
\eta(d) = 2^{-d(d+1)/2} \prod_{a=1}^{d} \frac{1 - q^{d-n-a}}{1 - 2^{-a}}
$$
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 $K$ is a random linear space of dimension $k$. We shall generate $K$ by repeatedly picking a random matrix $X \in \mathbb{F}_2^{d \times n}$ until $X$ has rank $d$ and then choosing $K = \ker(X)$. It is well-known that $X$ has rank $d$ with probability

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

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 $K = \ker(X)$.

The next step is computing $n \times n$ matrices $G$ and $G'$ such that $K$ is spanned by the first $k$ rows of $G$ and $GG^T = I$. Let us first set $K = \mathbb{F}_2^n$ and $G = \bar{G} = I$. Choose a zero quadratic form $q(x) = 0$ for all $x \in K$. Let $\xi^a$ be the $a$-th row of the matrix $X$. One can make $K$ orthogonal to $\xi^1, \ldots, \xi^d$ by making $d$ calls to the function SHRINK$^*$($\langle K, q \rangle, \xi^a, 0$) defined in Section IV B. (Recall that SHRINK$^*$ does not update the coefficients of $q$.) Finally we shift $K$ by a random uniformly distributed vector $h \in \mathbb{F}_2^n$. At this point $K$ is a random affine space represented in the standard form. It remains to choose random coefficients of the quadratic form $q : K \to \mathbb{Z}_8$ in the basis $g^1, \ldots, g^\ell$. Since $q$ must be distributed uniformly on the set of all quadratic forms $q : K \to \mathbb{Z}_8$, we must choose $Q \in \mathbb{Z}_8$. For $a < b$ as random uniform elements of the respective sets. Then the entire matrix $J$ is determined by $J_{a,b} = J_{b,a}$ and $J_{a,a} = 2D_{a,a}$, see Eq. (38). The entire algorithm is summarized below.

```plaintext
function RandomStabilizerState(n)
Compute $P(0), \ldots, P(n)$ from Eq. (72)
Sample $d \in \{0, 1, \ldots, n\}$ from $P(d)$
k ← $n - d$
repeat
Pick random $X \in \mathbb{F}_2^{d \times n}$
until rank($X$) = $d$
$G' \leftarrow I$, $G \leftarrow I$, $h \leftarrow 0^k$
$k \leftarrow (n, k, h, G, G')$
▷ Now $K = \mathbb{F}_2^n$ is full binary space
$q \leftarrow$ all-zeros function on $K$
for $a = 1$ to $d$
$\xi \leftarrow a$-th row of $X$
SHRINK$^*$($\langle K, q \rangle, \xi, 0$)
end for
▷ Now $K = \ker(X)$
▷ $K$ 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 $(n, k, h, G, G, Q, D, J)$
end function
```

Each call to SHRINK takes time $O(n^2)$, see Section IV B, whereas each computation of 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.

### E. Pauli measurements

Suppose $|K, q \rangle \in S_n$ is a stabilizer state of $n$ qubits represented in the standard form and $P \in P_n$ 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 $P_+|K, q \rangle$. 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 $\mathcal{G} \subseteq P_n$ 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 $\Pi_\mathcal{G}|K, q \rangle$ can be computed in time $O(n^3)$ using the above algorithm.

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

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

(73)
We shall consider two cases depending on whether or not \( \xi \in \mathcal{L}(K) \). This inclusion can be checked in time \( O(kn) \) by computing inner products \( \xi_a = (\xi, \bar{g}^a) \) with \( a = 1, \ldots, k \). Namely, \( \xi \in \mathcal{L}(K) \) iff \( \xi = \sum_{a=1}^{k} \xi_a g^a \pmod{2} \).

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

\[
\chi(x) = q(x + \xi) - q(x).
\]

By definition of a quadratic form one has

\[
\chi(h + y) = \chi(h) + J(\xi, y) \quad \text{for all } y \in \mathcal{L}(K).
\]

The state \( P_+|K, q \) can be written as

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

Perform a change of variable \( x = h \oplus y \) with \( y \in \mathcal{L}(K) \). Using Eq. (75) one can rewrite the above state as

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

with

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

and

\[
\lambda(y) = 4(\zeta, y) + J(\xi, y) \in \{0, 4\}.
\]

Let us first compute \( \omega \). We have

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

The decomposition Eq. (80) can be computed in time \( O(kn) \). Once the coefficients \( \xi_a \) are known, one can compute \( \omega \) from

\[
\omega = 2m + 4(\zeta, h) + \sum_{a=1}^{k} D_a \xi_a + \sum_{1 \leq a < b \leq k} J_{a,b} \xi_a \xi_b.
\]

This takes time \( O(kn) \).

Suppose first that \( \omega \in \{0, 4\} \). Then \( e^{i \frac{\pi}{2} \omega} = \pm 1 \) and thus

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

We get

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

where

\[
\mathcal{M} = \mathcal{K} \cap \{ x \in \mathbb{F}_2^n : \lambda(h \oplus x) = \omega \}.
\]

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

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

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

\[
4 \eta_a = \lambda(g^a) = 4(\zeta, g^a) + J(\xi, g^a), \quad 1 \leq a \leq k.
\]

To compute \( (\zeta, g^a) \) and \( J(\xi, g^a) \) consider expansions Eq. (80) and

\[
\zeta = \sum_{a=1}^{n} \zeta_a g^a \pmod{2}, \quad \zeta_a = (g^a, \zeta).
\]

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

\[
4 \eta_a = 4 \zeta_a + \sum_{b=1}^{k} J_{a,b} \zeta_b, \quad 1 \leq a \leq k.
\]

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. (82) can be computed by calling the function SHRINK(|K, q, \gamma, \alpha|), see Section IV B, which takes time \( O(kn) \).

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

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

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

\[
\lambda'(h \oplus y) = \begin{cases} 
0 \text{ if } \lambda(y) = 0, \\
2 \text{ if } \lambda(y) = 4.
\end{cases}
\]

Define

\[
\sigma = \begin{cases} 
1 \text{ if } \omega = 2, \\
-1 \text{ if } \omega = 6.
\end{cases}
\]

Then the state in Eq. (77) can be written as

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

with a quadratic form

\[
q'(x) = \sigma + q(x) - \sigma \lambda'(x).
\]

To get the standard form of \(|\mathcal{M}, q'\rangle \) we need to choose \( \lambda'(x) \) satisfying Eq. (88) and compute the coefficients of
\( \lambda'(x) \) in the basis \( g^1, \ldots, g^k \) of \( \mathcal{L}(K) \). First, let us compute the basis-dependent representation of \( \lambda(y) \). Suppose \( y = \sum_{a=1}^{k} y_ag^a \pmod{2} \) and let \( \bar{y} = (y_1, \ldots, y_k) \). Substituting Eqs. (80,86) into Eq. (79) one gets

\[
\lambda(\bar{y}) = 4 \sum_{a=1}^{k} \eta_ay_a,
\]

where \( \eta_a \in \{0,1\} \) are defined by Eq. (87). For any \( z_1, \ldots, z_k \in \{0,1\} \) one has the following identity:

\[
2(z_1 \oplus \cdots \oplus z_k) = 2 \sum_{a=1}^{k} z_a - 4 \sum_{1 \leq a < b \leq k} z_az_b \pmod{8}.
\]

Choose \( z_a = \eta_ay_a \) such that \( \lambda(\bar{y}) = 4(z_1 \oplus \cdots \oplus z_k) \). Then a function \( \lambda'(y) \) satisfying Eq. (88) has a basis-dependent representation \( \lambda'(\bar{y}) = 2(z_1 \oplus \cdots \oplus z_k) \), that is,

\[
\lambda'(h+y) = 2 \sum_{a=1}^{k} \eta_ay_a - 4 \sum_{1 \leq a < b \leq k} \eta_ay_by_b. \tag{92}
\]

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

\[
Q' = Q + \sigma, \quad D'_a = D_a - 2\sigma \eta_a, \tag{93}
\]

and

\[
J'_{a,b} = J_{a,b} + 4\eta_ay_by_b \quad \text{for} \ a \neq b. \tag{94}
\]

This determines the standard form of \( |K,q'\rangle \).

Case 2: \( \xi \notin \mathcal{L}(K) \). Then \( \xi \oplus x \notin \mathcal{K} \) for any \( x \in \mathcal{K} \) and thus the states \( |K,q'\rangle \) 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 \( \mathcal{L}(K) \) and \( \xi \). We equip \( \mathcal{L}(\mathcal{M}) \) with a basis \( g^1, \ldots, g^{k+1} \), where \( g^{k+1} = \xi \). Then any vector \( x \in \mathcal{M} \) can be written in a basis-dependent way as

\[
x = h + \sum_{a=1}^{k+1} x_ag^a \pmod{2}.
\]

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

\[
P_{+}|\mathcal{K},q'\rangle = 2^{-1-k/2} \sum_{x \in \mathcal{M}} e^{i\frac{\pi}{4}q'(x)|x}\rangle = 2^{-1/2}|\mathcal{M},q'\rangle, \tag{95}
\]

where \( \tilde{q} : \mathcal{M} \rightarrow \mathbb{Z}_8 \) is a quadratic form defined by

\[
\tilde{q}(\bar{x}) = q(\bar{x}) + [2m + 4(\zeta, h + \xi)]x_{k+1} + 4 \sum_{a=1}^{k} \zeta_ax_ax_{k+1}. \tag{96}
\]

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

\[
J' = \begin{bmatrix}
J & 4\zeta^T \\
4\zeta & 4m
\end{bmatrix} \tag{97}
\]

Here \( \zeta = (\zeta_1, \ldots, \zeta_k) \) is a row vector.

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

```python
function \text{M=EXTEND}(\mathcal{K}, \xi):
    S \leftarrow \{a \in [n] : (\xi, \bar{g}^a) = 1\}
    T \leftarrow S \setminus \{k+1, \ldots, n-1, n\}
    if T = \emptyset then
        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
    \text{Now } g^i = \xi
    Swap \bar{g}^i and \bar{g}^{k+1}. Swap \bar{g}^i and \bar{g}^{k+1}.
    return (n, k+1, h, G, G)
end function
```

It has runtime \( O(n^2) \). We do not have to update the coefficients of \( q' \) since Eq. (96) 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 \( \text{MeasurePauli} \) that takes as input a stabilizer state \( |K,q\rangle \in \mathcal{S}_n \) 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 $\Gamma = \text{MeasurePauli}([K, q], P)$
\begin{align*}
\text{\triangleright } & \quad P = i^n Z(\zeta) X(\xi) \\
\text{\triangleright } & \quad K = (n, k, h, G, \tilde{G}) \\
\text{\triangleright } & \quad q = (Q, D, J)
\end{align*}
\begin{algorithmic}
\For{$a = 1$ to $k$}
\State $\xi_a \leftarrow (g^a, \xi)$
\EndFor
\State $\xi' \leftarrow \sum_{a=1}^{k} \xi_a g^a \pmod{2}$
\State Compute $\omega \in \{0, 2, 4, 6\}$ using Eq. (81)
\If{$\xi' = \xi$ and $\omega \in \{0, 4\}$}
\State Compute $n_1, \ldots, n_k$ using Eq. (87)
\State $
\gamma \leftarrow \sum_{a=1}^{k} \eta_a g^a \pmod{2}$
\State $\omega' \leftarrow \omega/4$
\State $\alpha \leftarrow \omega' \pmod{\eta, h}$
\State $
\text{\triangleright } \text{SHRINK}([K, q], \gamma, \alpha)$
\If{$\epsilon = \text{EMPTY}$}
\State $\Gamma \leftarrow 0$
\Return
\EndIf
\If{$\epsilon = \text{SAME}$}
\State $\Gamma \leftarrow 1$
\Return
\EndIf
\EndIf
\EndIf
\If{$\xi' \neq \xi$}
\State $\text{\triangleright } \text{EXTEND}(K, \xi)$
\State $D \leftarrow [D, 2m + 4(\zeta, h \pmod{\xi}]]$
\State $J' \leftarrow J'$, where $J'$ is defined in Eq. (97)
\State $\Gamma \leftarrow 2^{-1/2}$
\Return
\EndIf
\EndIf
\EndFunction

F. Simulation of the hidden shift algorithm

Here we provide further details of the simulations reported in Fig. 1 of the main text. Recall that we simulate a circuit
\begin{equation}
U = H^\otimes n O_f H^\otimes n O_f H^\otimes n,
\end{equation}
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 $f, f' : F_2^n \rightarrow \{+1, -1\}$ such that
\begin{equation}
f'(x + s) = 2^{-n/2} \sum_{y \in F_2^n} (-1)^{x \cdot y} f(y) \quad \text{for all } x \in F_2^n.
\end{equation}

Here $s \in F_2^n$ is the hidden shift that can be found from $|s\rangle = U |0^\otimes n\rangle$. 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
\begin{equation}
g : F_2^n \rightarrow F_2 \quad \text{and} \quad \pi : F_2^{n/2} \rightarrow F_2^{n/2}
\end{equation}
be any Boolean function and any permutation respectively. For any such pair $g, \pi$ we may define a bent function $f : F_2^n \rightarrow \{+1, -1\}$ according to
\begin{equation}
f(x, y) = (-1)^{g(x) + y \cdot \pi(x)} \quad x, y \in F_2^n.
\end{equation}

The Hadamard transform of $f$ is given by
\begin{equation}
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))}.
\end{equation}

In our simulations we only used bent functions of the form Eq. (100) 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
\begin{equation}
O_g |x\rangle = (-1)^{g(x)} |x\rangle \quad x \in F_2^{n/2}
\end{equation}
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
\begin{equation}
O_f = \left( \prod_{i=1}^{n/2} CZ_{i, i+n/2} \right) O_g \otimes I
\end{equation}
where $CZ = \text{diag}(1, 1, 1, -1)$ 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. (99,101) one infers that
\begin{equation}
O_f' = X(s) \left[ \left( \prod_{i=1}^{n/2} CZ_{i, i+n/2} \right) I \otimes O_g \right] X(s)
\end{equation}

<table>
<thead>
<tr>
<th>Number of qubits</th>
<th>10</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MeasurePauli</td>
<td>0.27</td>
<td>0.3</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>RandomStabilizerState</td>
<td>0.2</td>
<td>0.3</td>
<td>0.8</td>
<td>1.7</td>
<td>2.8</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>InnerProduct</td>
<td>0.5</td>
<td>1.5</td>
<td>3.5</td>
<td>6.5</td>
<td>8.9</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>ExponentialSum</td>
<td>0.3</td>
<td>0.8</td>
<td>2.2</td>
<td>4.4</td>
<td>8</td>
<td>$O(n^3)$</td>
</tr>
</tbody>
</table>

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.
Note that the total $T$-count of the circuit $U$ is twice the $T$-count of $O_y$. To construct a circuit implementing $O_y$, 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 in the main text respectively), and then produced a circuit $O_y$ alternating the $CCZ$ gates (on a randomly chosen triple of qubits) with random sequences of 200 Clifford gates from the set $\{Z,C\}$. Note that the $CCZ$ gate can be replaced by the Toffoli gate using the identity

$$CCZ = (I \otimes I \otimes H)\text{Toff}(I \otimes I \otimes H).$$

(102)

To decompose Toffoli gadgets into Clifford and $T$-gates we used a gadget proposed by Jones [7], see Fig. 1. 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. (21, 22). Namely, in our simulation the measurement gadget uses four $T$-gates, two ancillary qubits (five and six for the simulations respectively), and then produced a circuit $O_y$ used a gadget proposed by Jones [7], see Fig. 1. The unitary $V_y$ is a $(n + 1 + TF + t)$-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 $|\psi\rangle$ in Eq. (103) is a $t$-qubit state which approximates $t$ copies of the magic state $|A^{\otimes t}\rangle$. In particular, $|\psi\rangle$ was derived from a $k$-dimensional subspace $\mathcal{L}$ of $F_2^k$ 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.

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

$$\langle 0^{\otimes N} \otimes \psi | V^\dagger_y (|1\rangle_0 \otimes |y\rangle_0) V_y | 0^{\otimes N} \otimes \psi \rangle = 2^{-u} \langle \psi | \Pi_\mathcal{F} | \psi \rangle$$

(104)

and

$$\langle 0^{\otimes N} \otimes \psi | V^\dagger_y (|0\rangle_0 \otimes |y\rangle_0) V_y | 0^{\otimes N} \otimes \psi \rangle = 2^{-v} \langle \psi | \Pi_\mathcal{G} | \psi \rangle$$

(105)

and then, if $\Pi_\mathcal{F} \neq 0$ and $\Pi_\mathcal{G} \neq 0$, we computed approximations $\alpha, \beta$ to the quantities Eqs. (104,105) 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 $P^\text{out}_y(1)$ was then $\alpha/(\alpha + \beta)$ (cf. Eq. (103)). Note that if either $\Pi_\mathcal{F} = 0$ or $\Pi_\mathcal{G} = 0$ then $\alpha, \beta$, and $P^\text{out}_y(1)$ can be computed without ever calling the norm estimation subroutine. This special case occurred for all qubits 1, 2, \ldots, 20 in both our simulations (as well as for some of the other data points).
