

Computing quopit Clifford circuit amplitudes by the sum-over-paths technique

Dax Enshan Koh,^{1,*} Mark D. Penney,^{2,†} and Robert W. Spekkens^{3,‡}

¹*Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

²*Max Planck Institute for Mathematics, Vivatsgasse 7, 53111 Bonn, Germany*

³*Perimeter Institute for Theoretical Physics, 31 Caroline Street North, Waterloo, Ontario, Canada N2L 2Y5*

By the Gottesman-Knill Theorem, the outcome probabilities of Clifford circuits can be computed efficiently. We present an alternative proof of this result for quopit Clifford circuits (i.e., Clifford circuits on collections of p -level systems, where p is an odd prime) using Feynman’s sum-over-paths technique, which allows the amplitudes of arbitrary quantum circuits to be expressed in terms of a weighted sum over computational paths. For a general quantum circuit, the sum over paths contains an exponential number of terms, and no efficient classical algorithm is known that can compute the sum. For quopit Clifford circuits, however, we show that the sum over paths takes a special form: it can be expressed as a product of Weil sums with quadratic polynomials, which can be computed efficiently. This provides a method for computing the outcome probabilities and amplitudes of such circuits efficiently, and is an application of the circuit-polynomial correspondence which relates quantum circuits to low-degree polynomials.

Keywords. Gottesman-Knill Theorem, quopit Clifford circuits, path integrals, circuit-polynomial correspondence

I. INTRODUCTION

Computing the outcome probabilities of a quantum circuit is in general a hard problem. In complexity-theoretic terms, it belongs to a class of problems known as $\#P$ -hard, which are widely conjectured to not be efficiently solvable by a classical computer (or even a quantum computer) [1]. Nevertheless, there are interesting subclasses of quantum circuits for which we do know efficient classical algorithms to compute the outcome probabilities. An example is the class of (nonadaptive) Clifford circuits, which has been studied extensively in quantum information theory, for example, in quantum error correction [2] and in measurement-based quantum computation [3, 4]. These circuits are rich enough to exhibit many of the ‘nonclassical’ features of quantum mechanics like entanglement and quantum teleportation, but yet are not rich enough to preclude efficient simulation by a classical computer [5]. The latter fact is the content of the Gottesman-Knill Theorem [6], and one of its implications is the existence of an efficient classical algorithm to compute the outcome probabilities of a Clifford circuit.

The original proof of the Gottesman-Knill Theorem makes use of the *stabilizer* formulation of quantum mechanics, in which the state of the system at each time step is represented not by the amplitudes of the state vector, but by a set of Pauli operators which stabilize it [2]. Using this approach, the problem of computing the outcome probabilities of Clifford circuits can be reduced to computing inner products between stabilizer states [5]. The latter can be done efficiently using the stabilizer formalism, and hence the outcome probabilities can be com-

puted efficiently.

Besides the stabilizer formalism, other techniques have been used to compute the outcome probabilities of Clifford circuits efficiently (for some examples, see [7–10]). In this paper, we present a different method from these that is explicitly based upon Feynman’s sum-over-paths technique [11–13]. We restrict our attention to Clifford circuits acting on collections of quopits, i.e., p -level systems where p is an odd prime [14] (a few remarks about extending our results to qubit systems will be made in Section VII). In this approach, the amplitudes of quantum circuits are expressed in terms of a weighted sum over computational paths.

For general quantum circuits, such a sum over paths involves an exponential number of terms, and no efficient algorithm exists to compute this sum, unless $\#P$ -complete problems can also be solved efficiently. However, building on the work of Dawson *et al.* [12], we show that for quopit Clifford circuits, the sum over paths takes a special form: it can be expressed as a product of Weil sums [15] with quadratic polynomials. The problem of evaluating Weil sums explicitly is in general difficult, but for Weil sums with quadratic polynomials, the sum can be computed efficiently. This gives an efficient algorithm to compute not just the outcome probabilities but also the amplitudes of quopit Clifford circuits when all n quopit registers are measured. In other words, such circuits admit of an efficient $STR(n)$ simulation [16] (see also Section V for a discussion of various notions of simulation).

The sum-over-paths technique has previously been used to answer computational complexity questions about the power of quantum computation. For example, by considering quantum circuits comprising only gates from the universal gate set of Toffoli and Hadamard gates, Dawson *et al.* provide a simple proof of the complexity-theoretic result that $BQP \subseteq PP$ (first proved by [17]), one of the tightest ‘natural’ upper bounds for BQP [12]. Dawson *et al.* then ask what other universal

*Electronic address: daxkoh@mit.edu

†Electronic address: mpenney@mpim-bonn.mpg.de

‡Electronic address: rspekkens@perimeterinstitute.ca

gate sets are amenable to the sum-over-paths approach. An extension of this question, that we address in this paper, is to ask not just about universal gate sets, but also about gate sets corresponding to restricted models of quantum computation.

Another example is the class of linear algebraic quantum circuits (which are closely related to Clifford circuits) studied by Bacon *et al.* [18], who noted that the sum-over-paths technique introduced by Dawson *et al.* implied that the computation of outcome probabilities in such circuits (assuming all registers are measured) can be reduced to the computation of Weil sums for quadratic polynomials, implying efficient classical simulation of such circuits (specifically, efficient STR(n) simulation). When specialized to the case of quopits, however, the group of unitaries implementable in a linear algebraic quantum circuit is a proper subgroup of those implementable by a quopit Clifford circuit because the generating gate set does not include the phase gate (R in Eq. (2), which corresponds to a phase space squeezing operation). In this respect, our result generalizes theirs. Furthermore, we here provide an explicit expression for not just the outcome probabilities, as Bacon *et al.* do, but the amplitudes as well.

The sum-over-paths technique makes explicit a correspondence between quantum circuits and low-degree polynomials, known as the *circuit-polynomial correspondence* [19]. This correspondence can be exploited in two different directions. In the first direction, using quantum circuit concepts, it enables one to prove classical results about polynomials. For example, the Gottesman-Knill Theorem, which is a theorem about quantum circuits, can be used to provide an efficient algorithm to compute the gap of degree-2 polynomials over \mathbb{F}_2 [19]. In the second direction, known classical results about polynomials can be used to provide algorithms for simulating quopit Clifford circuits, provides an example of the second direction. Note that while the polynomials in [12] and [19] are over \mathbb{F}_2 , our results about quopit systems involve polynomials over the field \mathbb{F}_p where p is an odd prime.

The rest of the paper is structured as follows. In Section II, we introduce the relevant definitions and notations and describe the problem of interest. In Section III, we review the sum-over-paths technique and show how to construct sum-over-paths expressions for quopit Clifford circuits. In Section IV, we show how the sum-over-paths expression can be computed classically in polynomial time. In Section V, we discuss different notions of classical simulation and their relation to the Gottesman-Knill Theorem. In Section VI, we show how our results can be used to show that unitary operations implemented by quopit Clifford circuits are necessarily balanced. Finally, in Section VII, we conclude by discussing other gate sets, including qubit Clifford gates.

II. PRELIMINARY DEFINITIONS AND NOTATION

In this paper, p will always denote an odd prime. We shall work over the finite field \mathbb{F}_p of characteristic p , which is the set of integers modulo p . The set of $n \times n$ matrices over \mathbb{F}_p is denoted by $M_n(\mathbb{F}_p)$, and the group of invertible $n \times n$ matrices over \mathbb{F}_p is denoted by $GL_n(\mathbb{F}_p)$.

We confine our attention to quopit systems, i.e., p -level quantum systems where p is an odd prime. A quopit Clifford circuit acting on quopit systems is defined to be any circuit consisting of only the following gates, called *quopit Clifford gates*: the Fourier gate F , the phase gate R and the sum gate Σ , which are the p -level generalizations, respectively, of the Hadamard, phase and CNOT gates of qubit Clifford circuits defined in Eq. (14). They are defined as follows:

$$\begin{aligned} F &\equiv \frac{1}{\sqrt{p}} \sum_{s,t \in \mathbb{F}_p} \chi(st) |s\rangle \langle t|, \\ R &\equiv \sum_{t \in \mathbb{F}_p} \chi(t(t-1)2^{-1}) |t\rangle \langle t|, \\ \Sigma &\equiv \sum_{s,t \in \mathbb{F}_p} |s, s+t\rangle \langle s, t|, \end{aligned} \quad (2)$$

where $\chi(a) \equiv \exp(2\pi ia/p)$, and $2^{-1} = (p+1)/2$ is the inverse of 2 modulo p . For the sum gate, we write Σ_{ab} to indicate that a and b are the control and target registers respectively, i.e. $\Sigma_{ab} = \sum_{s,t \in \mathbb{F}_p} |s\rangle \langle s|_a \otimes |s+t\rangle \langle t|_b$.

For a given circuit, let n denote the number of registers (i.e. number of quopits), and N denote the number of gates. We make the following additional assumptions about the circuit (for an example, see the circuit diagram in Figure 1):

- The inputs to the circuit are computational basis states $|a\rangle$, where $a \in \mathbb{F}_p^n$.
- Measurements are performed only at the end of the circuit, i.e., there are no intermediate measurements, and *all* quopits are measured at the end of the circuit. Also, measurements are performed in the computational basis. Hence, the possible measurement outcomes lie in the set \mathbb{F}_p^n . A measurement outcome of $b \in \mathbb{F}_p^n$ is associated with the computational basis vector $|b\rangle$.
- There are no extraneous quopits, i.e. every quopit is acted on by at least one gate, so that $n = O(N)$.

The problem we are interested in, which we call \mathcal{P} , is the following: given a quopit Clifford circuit acting on the input state $|a\rangle$, where $a \in \mathbb{F}_p^n$, compute the probability amplitude associated with the outcome $b \in \mathbb{F}_p^n$. Formally, \mathcal{P} may be stated as follows:

Given a description of a quopit Clifford circuit that implements the unitary U , as well as strings $a, b \in \mathbb{F}_p^n$,

compute $\langle b|U|a\rangle$.

Note that a description of a quopit Clifford circuit C is a specification of the gates in C as well as the registers on which they act.

If C were allowed to be a general quantum circuit with gates chosen from some universal discrete gate set, then the problem \mathcal{P} would be $\#P$ -hard. But for the quopit Clifford circuits C that we consider, \mathcal{P} can be solved in polynomial-time. We now describe a proof of this result that is based on the sum-over-paths formulation of quopit Clifford circuits.

III. CONSTRUCTING SUM-OVER-PATHS EXPRESSIONS FOR QUOPIT CLIFFORD CIRCUITS

In this section, we review the sum-over-paths technique applied to quopit Clifford circuits that was introduced in Section III of Ref. [13]. Without loss of generality, we assume that each register of the Clifford circuit terminates in a Fourier gate just before it is measured (we shall refer to circuits with this property as *standard-form quopit Clifford circuits*). If this were not the case, for each register that does not terminate in a Fourier gate, we could pad the circuit by inserting 4 Fourier gates before the measurement is performed, since $F^4 = \mathbb{I}$. The Fourier gates that appear just before a measurement shall be called *terminal* Fourier gates. All other Fourier gates will be called *non-terminal*.

For a quopit Clifford circuit C with input labeled by $a = a_1 \dots a_n \in \mathbb{F}_p^n$ and measurement outcome labeled by $b = b_1 \dots b_n \in \mathbb{F}_p^n$, we shall label wires of C at every time step to create a *labeled circuit* as follows (See Figures 1 and 2 for an example):

1. Label the input wires by a_1, \dots, a_n .
2. Going from left to right of the circuit diagram for C , label the wires at each subsequent time step as follows:
 - (a) For each phase gate R and identity gate \mathbb{I} (i.e. when we have a bare wire), if the label at the input is s , then we label the output by s .
 - (b) For each sum gate Σ , if the labels at the inputs are (s, t) , then label the outputs by $(s, s + t)$. Here the first element in the pair is the control register, and the second element in the pair is the target register.
 - (c) For the l th non-terminal Fourier gate F , we introduce an auxiliary variable x_l , and regardless of the input to the Fourier gate, we label the output by x_l .
3. Label the output wires by b_1, \dots, b_n .

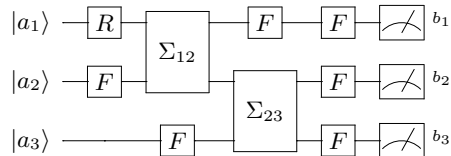


FIG. 1: Example of a quopit Clifford circuit. As explained in the text, we can assume without loss of generality that each register ends in a Fourier gate.

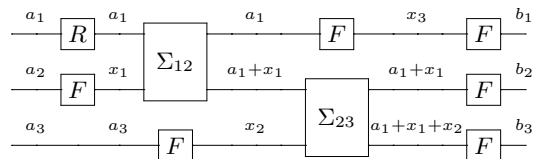


FIG. 2: Labeled circuit corresponding to the circuit in Figure 1. The phase polynomial is read off to be $S(x_1, x_2, x_3) = a_2x_1 + a_3x_2 + a_1x_3 + x_3b_1 + b_2(a_1 + x_1) + b_3(a_1 + x_1 + x_2) + 2^{-1}a_1(a_1 - 1)$.

We shall associate each quopit Clifford circuit with a polynomial over \mathbb{F}_p , which is called the *phase polynomial* [20]. The variables in the phase polynomial are the auxiliary variables $x = (x_1, \dots, x_\alpha)$, where α is the number of non-terminal Fourier gates in the circuit. For each gate G in the circuit, let $\text{in}(G)$ and $\text{out}(G)$ be the input and output labels of that gate in the labeled circuit. The phase polynomial associated with a quopit Clifford circuit is the polynomial S over \mathbb{F}_p defined by (see Figure 2 for an example):

$$S(x) = \sum_{\text{Fourier gates } F} \text{in}(F)\text{out}(F) + \sum_{\text{phase gates } R} 2^{-1} \text{in}(R)(\text{in}(R) - 1). \quad (3)$$

The theorem relating the circuit amplitudes and the phase polynomial, which appears as Theorem 3 in [13], is the following:

Theorem 1. *Let C be a standard-form quopit Clifford circuit on n registers that implements the Clifford operation U . Let α be the number of non-terminal Fourier gates and let $S(x)$ be the phase polynomial associated with C . Then,*

$$\langle b|U|a\rangle = \frac{1}{p^{(n+\alpha)/2}} \sum_{x \in \mathbb{F}_p^\alpha} \chi(S(x)). \quad (4)$$

IV. EVALUATING THE SUM OVER PATHS

Using Theorem 1, the problem \mathcal{P} is reduced to evaluating the sum in Eq. (4). In this section, we describe how this sum may be evaluated.

First, we note that $S(x)$ is a degree-2 polynomial in the variables $x = (x_1, \dots, x_\alpha)$ as well as the variables $a_1, \dots, a_n, b_1, \dots, b_n$. This is due to the fact that $S(x)$ is a sum of terms which are at most quadratic, since the input and output labels of each gate are linear in the variables x_i, a_i and b_i . Hence, we can write $S(x)$ as

$$S(x) = x^T \Theta x + \eta^T x + \zeta = \sum_{i,j=1}^{\alpha} \Theta_{ij} x_i x_j + \sum_{i=1}^{\alpha} \eta_i x_i + \zeta, \quad (5)$$

where $\Theta \in M_\alpha(\mathbb{F}_p)$ can be chosen to be symmetric, $\eta \in \mathbb{F}_p^\alpha$ and $\zeta \in \mathbb{F}_p$. Note that while η and ζ are dependent on a and b , Θ is independent of (a, b) , because otherwise $S(x)$ as a polynomial in x_i, a_i and b_i would have a degree that exceeds 2.

Substituting Eq. (5) into Eq. (4) gives

$$\langle b|U|a \rangle = \frac{\chi(\zeta)}{p^{(n+\alpha)/2}} \sum_{x \in \mathbb{F}_p^\alpha} \chi(x^T \Theta x + \eta^T x). \quad (6)$$

The above sum can be evaluated using the following two steps.

A. Step 1: Diagonalizing Θ

In the first step, we diagonalize the matrix Θ , by making use of the following theorem:

Theorem 2. *There is a polynomial-time algorithm T that when given a symmetric matrix $\Theta \in M_\alpha(\mathbb{F}_p)$ outputs an invertible matrix $L \in \text{GL}_\alpha(\mathbb{F}_p)$ such that $L^T \Theta L$ is diagonal.*

Proof. The proof is essentially an algorithmic implementation of the standard proof that any quadratic form over a field that is not of characteristic 2 is diagonalizable (see Propositions 6.20 and 6.21 of [21]). We present a polynomial-time algorithm in Appendix A. \square

By making use of Theorem 2, and the change of variables $\mu = L^T \eta$ and $x = Ly$, we can rewrite $x^T \Theta x + \eta^T x = y^T \Lambda y + \mu^T y$, where $\Lambda = L^T \Theta L$ is a diagonal matrix. By this change of variables, the sum in Eq. (6) becomes

$$\begin{aligned} \sum_{x \in \mathbb{F}_p^\alpha} \chi(x^T \Theta x + \eta^T x) &= \sum_{y_1, \dots, y_\alpha \in \mathbb{F}_p} \chi \left(\sum_{i=1}^{\alpha} \lambda_i y_i^2 + \mu_i y_i \right) \\ &= \prod_{i=1}^{\alpha} \sum_{y_i \in \mathbb{F}_p} \chi(\lambda_i y_i^2 + \mu_i y_i), \end{aligned} \quad (7)$$

where the λ_i are the diagonal entries of Λ , and the μ_i are the components of μ . We see from Eq. (7) that one needs only to compute the (much simpler) sums over a single variable. Such sums are called *Weil sums* [21], and we will show in the next step how to compute them.

B. Step 2: Using the exponential sum formula

The second step makes use of the following theorem about exponential sums (see Theorem 5.33 of [21]):

Theorem 3. *The sum*

$$\sum_{y \in \mathbb{F}_p} \chi(\lambda y^2 + \mu y)$$

can be explicitly evaluated as follows:

1. *If $\lambda = \mu = 0$, then it equals p .*
2. *If $\lambda = 0$ and $\mu \neq 0$, then it equals 0.*
3. *If $\lambda \neq 0$, then it equals*

$$i^{\varepsilon(p)} \chi(-4^{-1} \lambda^{-1} \mu^2) \left(\frac{\lambda}{p} \right) \sqrt{p}, \quad (8)$$

where $\left(\frac{\lambda}{p} \right)$ is the Legendre symbol and $\varepsilon(p) = 0$ if p is congruent to 1 mod 4 and $\varepsilon(p) = 1$ otherwise. The inverses are modular multiplicative inverses modulo p .

If we partition the indices $i \in \{1, \dots, \alpha\}$ into the following sets

$$\begin{aligned} X &= \{i \in \{1, \dots, \alpha\} | \lambda_i \neq 0\}, \\ Y &= \{i \in \{1, \dots, \alpha\} | \lambda_i = 0, \mu_i = 0\}, \\ Z &= \{i \in \{1, \dots, \alpha\} | \lambda_i = 0, \mu_i \neq 0\}, \end{aligned}$$

then by using the exponential sum formula in Theorem 3 to evaluate the sums in Eq. (7), we obtain

$$\begin{aligned} \langle b|U|a \rangle &= \frac{\chi(\zeta)}{p^{(n+\alpha)/2}} \left(\prod_{i \in X} i^{\varepsilon(p)} \chi(-4^{-1} \lambda_i^{-1} \mu_i^2) \left(\frac{\lambda_i}{p} \right) \sqrt{p} \right) \\ &\quad \times \left(\prod_{j \in Y} p \right) \left(\prod_{k \in Z} 0 \right) \\ &= p^{-(n+r-\alpha)/2} \delta_{0,|Z|} i^{r\varepsilon(p)} \left(\frac{\prod_{i \in X} \lambda_i}{p} \right) \\ &\quad \times \chi \left(\zeta - 4^{-1} \sum_{i \in X} \lambda_i^{-1} \mu_i^2 \right), \end{aligned} \quad (9)$$

where $r = |X|$ is the rank of Θ , i.e. the number of nonzero diagonal entries in Λ . Note that we used the multiplicative property of the Legendre symbol: $\left(\frac{\prod_{i \in X} \lambda_i}{p} \right) = \prod_{i \in X} \left(\frac{\lambda_i}{p} \right)$, and the fact that when $|Z| = 0$, $|X| + |Y| = \alpha$. Here, $\delta_{x,y}$ is the Kronecker delta.

Now, the Legendre symbol $\left(\frac{a}{p} \right)$ takes values in the set $\{-1, 0, 1\}$ and vanishes only when $a \equiv 0 \pmod{p}$. Since $\lambda_i \neq 0$ for $i \in X$, by definition, we get the following simple expression for the outcome probabilities:

$$|\langle b|U|a \rangle|^2 = \frac{1}{p^{n+r-\alpha}} \delta_{0,|Z|}. \quad (10)$$

C. Running time

We now analyze the running time of the above procedure. The evaluation of the matrix element $\langle b|U|a\rangle$ involved four main steps: First, given a decomposition of U in terms of quopit Clifford gates, we employed the labeling procedure described in Section III to label the Clifford circuit. Second, from the labeled circuit, we computed the phase polynomial $S(x)$ defined by Eq. (3). Third, we used Theorem 2 to diagonalize $S(x)$, and fourth, we used the exponential sum formula in Theorem 3 to calculate the matrix element in Eq. (9).

Steps 1, 2, and 4 take time that is linear in the size of the circuit. Step 3 involves matrix diagonalization which can be carried out in polynomial time, as we show in Appendix A. Hence, the algorithm that we give here to compute $\langle b|U|a\rangle$ runs in polynomial time.

V. NOTIONS OF CLASSICAL SIMULATION AND THE GOTTESMAN-KNILL THEOREM

The Gottesman-Knill Theorem is a result stating that Clifford circuits can be efficiently simulated by a classical computer. Since its first appearance in [6], several other variants and generalizations of the theorem have been found, and today the term is often used, loosely, to refer to any one of a collection of results that are variants of the original theorem [9, 16, 22–24]. These variants vary according to the ingredients of the Clifford circuit and what it means to simulate it. The goal of this section is to clarify some of the differences between these variants and to discuss the relationship between the Gottesman-Knill Theorem and our results from Section IV that were obtained via the sum-over-paths approach.

Two common notions of classical simulation of quantum computation are *weak* and *strong* simulation [8, 25]. Let T be a description of a quopit quantum circuit with n registers, which we index by the integers in $[n] = \{1, \dots, n\}$. Let $I = \{i_1, \dots, i_{|I|}\} \subseteq [n]$ be a subset of indices, and let $y_{|I|} \in \mathbb{F}_p^{|I|}$ be a $|I|$ -tuple of elements from \mathbb{F}_p . Define $p_T^I(y_{|I|})$ to be the probability that the outcomes $y_{|I|}$ are observed when the registers whose indices are in the set I of the quantum circuit T are measured. A strong simulation of a family of quopit circuits is a deterministic classical algorithm that takes as input a triple $\langle T, I, y_{|I|} \rangle$ and outputs the probability $p_T^I(y_{|I|})$. A weak simulation of a family of quopit circuits is a randomized classical algorithm that takes as input the pair $\langle T, I \rangle$ and outputs the values $y \in \mathbb{F}_p^{|I|}$ according to the probability distribution p_T^I . Stated informally, a strong simulation involves computing the marginal probabilities of the measurement outcomes of a quantum circuit, while a weak simulation involves just sampling from the same distribution as the quantum circuit. It was shown in [25] that strong simulation implies weak simulation.

Note that for the notions of strong and weak simula-

tions, no restrictions are placed on the size of the subsets I that are fed as inputs to the algorithm. To describe finer-grained notions of simulation that take into account the size of the subset of outputs to be simulated, we adopt the terminology introduced by Koh [16]: Let $f(n)$ be a function of the number of registers n in the circuit. Similar to strong simulation, a $\text{STR}(f(n))$ -simulation (which stands for strong- $f(n)$ simulation) of a family of quopit circuits is a deterministic classical algorithm that takes as input a triple $\langle T, I, y \rangle$ and outputs the probability $p_T^I(y)$, with the restriction that the subset $I = \{i_1, \dots, i_{f(n)}\} \subseteq [n]$ and the tuple $y \in \mathbb{F}_p^{f(n)}$ must be of size $f(n)$. Similarly, a $\text{WEAK}(f(n))$ -simulation is defined the same way as weak simulation, except that the subset I is required to be of size $f(n)$. The class that is relevant to our results is $\text{STR}(n)$, which is obtained by taking $f(n) = n$. While an efficient strong simulation implies an efficient $\text{STR}(n)$ -simulation, the latter seems incomparable to efficient weak simulation [16].

The original Gottesman-Knill Theorem states that there exists an efficient weak simulation of adaptive qubit Clifford circuits with computational basis inputs and computational basis measurements. Here, an efficient simulation is one that can be carried out in polynomial-time, and an adaptive circuit is one which has intermediate measurements whose outcomes may affect which operations we perform next. By changing the ingredients of the Clifford circuit as well as considering different functions $f(n)$ in the definitions of $\text{STR}(f(n))$ and $\text{WEAK}(f(n))$ simulations, several variants of the Gottesman-Knill Theorem can be obtained [9, 16]. Not all variations of ingredients and notions of simulation lead to efficient classical simulability though. In fact, the situation is much more delicate. As observed by Jozsa and Van den Nest [9] and Koh [16], small changes to the ingredients of a Clifford circuit can lead to dramatic changes in the simulation complexity. For example, while non-adaptive qubit Clifford circuits with computational basis inputs and computational basis measurements are efficiently strongly simulable, adaptive qubit Clifford circuits with all the other ingredients kept the same are unlikely to be efficiently strongly simulable (an efficient strong simulation of such circuits can be shown to be $\#P$ -hard [9]), even though it might seem that adaptability is a *classical* resource. A table classifying which combinations of ingredients and which notions of simulations lead to efficient simulations is presented in [16] (which generalizes a similar table found in [9]).

Using the terminology described above, our main result may be stated as follows.

Theorem 4. *There exists an efficient $\text{STR}(n)$ -simulation of nonadaptive quopit Clifford circuits with computational basis inputs and computational basis measurements. In particular, the amplitude associated with starting with a computational basis state $|a\rangle$ as the input to the circuit, where $a \in \mathbb{F}_p$, and measuring the result $b \in \mathbb{F}_p$, is given by Eq. (9), and the corresponding proba-*

bility is given by Eq. (10).

A few remarks are in order. First, we note that Theorem 4 corresponds to Case (iv) of Table 1 in [16], if the results in the paper are extended to quopit Clifford circuits. Secondly, it is unlikely that we can extend the theorem to include adaptive quopit Clifford circuits, since an analogous theorem to Theorem 2 of [16], which states that STR(n)-simulation of adaptive Clifford circuits is #P-hard, should hold true for quopit Clifford circuits. Thirdly, we leave open the question about whether the sum-over-paths approach is useful for proving efficient STR(1) or strong simulation of quopit Clifford circuits (we know that these exist though, by the Gottesman-Knill Theorem). Note that Theorem 4 does not immediately imply either efficient STR(1) or efficient strong simulation since, in general, computing a marginal probability from a joint probability requires summing an exponential number of terms and cannot be performed efficiently, unless there is some structure in the problem.

VI. BALANCEDNESS OF QUOPIT CLIFFORD CIRCUITS

Quopit Clifford gates have the property that their nonzero matrix elements relative to the computation basis have the same absolute value. A gate with this property (and the matrix representing it) is called *balanced* [12, 13].

Definition 1. A gate G acting on n qudits represented by a unitary U_G is balanced if there is a constant $c \in \mathbb{R}_{\geq 0}$ and functions $f : (\mathbb{Z}_d)^n \times (\mathbb{Z}_d)^n \rightarrow \mathbb{R}$ and $g : (\mathbb{Z}_d)^n \times (\mathbb{Z}_d)^n \rightarrow (\mathbb{Z}_d)^n$ such that for all $a, b \in (\mathbb{Z}_d)^n$,

$$\langle b|U_G|a\rangle = c e^{if(a,b)} \delta_{0,g(a,b)}. \quad (11)$$

As noted in [13], only if all gates in a circuit are balanced can one use the sum-over-paths technique to evaluate its functionality. We shall refer to the number c as the *weight* of the gate G . By convention, whenever the basis is not specified, it is assumed that the balanced property is defined with respect to the computational basis. Hence, the Fourier, phase and sum gates defined in Eq. (2) are balanced with weights $p^{-1/2}$, 1 and 1 respectively.

Unitary operations implemented by circuits consisting of balanced gates are not balanced in general. For example, consider a circuit consisting of the Hadamard gate H , given by (14), and the gate V , given by

$$V = \frac{1}{\sqrt{2}} \begin{pmatrix} e^{-i\theta} & -e^{i\theta} \\ e^{-i\theta} & e^{i\theta} \end{pmatrix}.$$

It is straightforward to check that both H and V are balanced and unitary. The product VH , however, is

$$VH = \begin{pmatrix} -i \sin(\theta) & \cos(\theta) \\ \cos(\theta) & -i \sin(\theta) \end{pmatrix},$$

which is not balanced in general. For example, when $\theta = \pi/6$, the entries of VH have absolute values $1/2$ and $\sqrt{3}/2$.

For quopit Clifford circuits, however, unitary operations implemented by quopit Clifford gates are always balanced. This is a direct consequence of Eq. (10). To see this, recall that Θ (defined in Eq. (5)) is independent of (a, b) . Hence, $r = \text{rank}(\Theta)$ is independent of (a, b) , which implies that the nonzero terms of $|\langle b|U|a\rangle|$, which are equal to $p^{-(n+r-\alpha)/2}$, are independent of (a, b) . This result is summarized in the following theorem:

Theorem 5. Let U be a unitary operation on n quopits that is implemented by a standard-form Clifford circuit C with α non-terminal Fourier gates, and phase polynomial $S(x)$. Let r be the rank of the coefficient matrix of the quadratic form corresponding to the degree-2 terms in $S(x)$. Then U is a balanced matrix with weight $p^{-(n+r-\alpha)/2}$.

VII. CONCLUDING REMARKS

The sum-over-paths approach has proved useful in providing an efficient algorithm for computing the amplitudes of circuits composed of quopit Clifford gates. The approach, however, is by no means applicable to only quopit Clifford circuits – our proof of Theorem 4 can easily be extended to imply a stronger result. To see this, note that the arguments presented in the proof depended on only the following two properties of the set \mathcal{S} of quopit Clifford gates:

1. Each gate $G \in \mathcal{S}$ is balanced with the function $g(a, b)$ being linear and the function $f(a, b)$ having the form

$$f(a, b) = \frac{2\pi}{p} S_G(a, b), \quad (12)$$

where S_G is a polynomial of degree at most 2 in the components of a and b having coefficients in \mathbb{F}_p .

2. There is a gate $G \in \mathcal{S}$ with finite order (i.e. there exists a natural number k such that $(U_G)^k = \mathbb{I}$) that satisfies $g(a, b) = 0$ for all a, b .

For any class of circuits composed of gates satisfying these two properties, the transition amplitudes can be computed by a sum-over-paths expression of the form

$$\mathcal{C} \sum_{x \in \mathbb{F}_p^\beta} \chi(S(x)), \quad (13)$$

where $\mathcal{C} \in \mathbb{R}_{>0}$ and $\beta \in \mathbb{N}$ are constants determined by the particular circuit, and $S(x)$ is a degree-2 polynomial in β variables having coefficients in \mathbb{F}_p . Theorem 3 then gives us an efficient STR(n)-simulation for such a class of circuits. In particular this includes the linear algebraic quantum circuits of Bacon *et al.* [18].

We conclude with a discussion about circuits with gate sets that do not satisfy the above two properties. Two classes of such circuits are of interest here. The first class of circuits are those composed of gates from universal gate sets. For example, consider a circuit composed of Toffoli and Hadamard gates. In general, the phase polynomial corresponding to such circuits is of degree 3, and hence does not satisfy Property 1 – the sum-over-paths technique that leads to efficient simulation presented in this paper does not apply to such circuits. Based on conjectures in computational complexity theory, this result is expected since it is believed that an efficient classical simulation of (universal) quantum circuits is not possible. The second class of circuits are efficiently simulable circuits like the *qubit Clifford circuits*, which are circuits composed of the Hadamard gate H , the phase gate P , and CNOT gate, defined as follows:

$$\begin{aligned} H &= \frac{1}{\sqrt{2}} \sum_{s,t \in \mathbb{F}_2} (-1)^{st} |s\rangle\langle t|, \\ P &= \sum_{t \in \mathbb{F}_2} i^t |t\rangle\langle t|, \\ \text{CNOT} &= \sum_{s,t \in \mathbb{F}_2} |s, s+t\rangle\langle s, t|. \end{aligned} \quad (14)$$

From the above expression for the phase gate P , it is ev-

ident that we cannot find a polynomial $S_P(s, t)$ over \mathbb{F}_2 such that the matrix elements $\langle s|P|t\rangle$ that are nonzero are of the form $\chi(S_P(s, t))$. As discussed in [13], one needs to define the phase polynomial to be over \mathbb{Z}_4 instead. We note also that for qubit Clifford circuits, there are other parts of the proof which would fail, for example, Theorem 2, which works only for fields that are not of characteristic 2. For a treatment of evaluating amplitudes of qubit Clifford circuits using exponential sums, we refer the reader to [26].

Acknowledgements

DEK thanks Siong Thye Goh for helpful discussions. DEK is supported by the National Science Scholarship from the Agency for Science, Technology and Research (A*STAR). MDP is supported by NSERC through the Doctoral Postgraduate Scholarship and by Corpus Christi College, Oxford. Part of this research was conducted during the 2015 Convergence Conference at the Perimeter Institute for Theoretical Physics. Research at the Perimeter Institute for Theoretical Physics is supported in part by the Government of Canada through NSERC and by the Province of Ontario through MRI.

-
- [1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- [2] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997.
- [3] Robert Raussendorf, Daniel E. Browne, and Hans J. Briegel. Measurement-based quantum computation on cluster states. *Phys. Rev. A*, 68:022312, Aug 2003.
- [4] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.
- [5] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.
- [6] Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.
- [7] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over $\text{GF}(2)$. *Phys. Rev. A*, 68:042318, Oct 2003.
- [8] M. Van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation*, 10(3-4):0258–0271, 2010.
- [9] Richard Jozsa and Maarten Van Den Nest. Classical simulation complexity of extended Clifford circuits. *Quantum Info. Comput.*, 14:633–648, May 2014.
- [10] Maarten Van den Nest. Efficient classical simulations of quantum Fourier transforms and normalizer circuits over Abelian groups. *arXiv preprint arXiv:1201.4867*, 2012.
- [11] Richard P. Feynman and A. R. Hibbs. *Quantum Mechanics and Path Integrals*. McGraw-Hill Companies, 1965.
- [12] Christopher M. Dawson, Henry L. Haselgrove, Andrew P. Hines, Duncan Mortimer, Michael A. Nielsen, and Tobias J. Osborne. Quantum computing and polynomial equations over the finite field \mathbb{Z}_2 . *Quantum Information & Computation*, 5(2):102–112, 2005.
- [13] Mark D Penney, Dax Enshan Koh, and Robert W Spekkens. Quantum circuit dynamics via path integrals: Is there a classical action for discrete-time paths? *New Journal of Physics*, 19(7):073006, 2017.
- [14] J. Emerson, V. Veitch, M. Howard, D. Gottesman, A. Hamed, C. Ferrie, and D. Gross. Negative quasiprobability, contextuality, quantum magic and the power of quantum computation, 2012. Slides of a talk given at UBC, July 2013.
- [15] André Weil. Numbers of solutions of equations in finite fields. *Bull. Amer. Math. Soc.*, 55:497–508, 1949.
- [16] Dax Enshan Koh. Further extensions of Clifford circuits and their classical simulation complexities. *Quantum Information & Computation*, 17(3&4):0262–0282, 2017.
- [17] Leonard M Adleman, Jonathan Demarrais, and Ming-Deh A Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.
- [18] D. Bacon, W. Van Dam, and A. Russell. Analyzing algebraic quantum circuits using exponential sums. *unpublished*. Available at <http://www.cs.ucsb.edu/vandam/publications.html>, 2008.
- [19] Ashley Montanaro. Quantum circuits and low-degree

- polynomials over \mathbb{F}_2 . *arXiv preprint arXiv:1607.08473*, 2016.
- [20] Vladimir P Gerdt and Vasily M Severyanov. An algorithm for constructing polynomial systems whose solution space characterizes quantum circuits. In *Quantum Informatics 2005*, pages 626406–626406. International Society for Optics and Photonics, 2006.
- [21] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, 1997.
- [22] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.
- [23] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.
- [24] Maarten Van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation*, 10(3-4):258–271, 2010.
- [25] Barbara M Terhal and David P DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games. *Quantum Information & Computation*, 4(2):134–145, 2004.
- [26] Sergey Bravyi and David Gosset. Improved classical simulation of quantum circuits dominated by Clifford gates. *Phys. Rev. Lett.*, 116(25):250501, 2016.

Appendix A: Proof of Theorem 2

We shall describe a polynomial-time algorithm T that, when given a symmetric matrix $\Theta \in M_\alpha(\mathbb{F}_p)$, outputs an invertible matrix $L \in \text{GL}_\alpha(\mathbb{F}_p)$ such that $L^T \Theta L$ is diagonal. As we assumed in the main text, p denotes an odd prime. The proof is essentially an algorithmic implementation of Propositions 6.20 and 6.21 of [21].

We use the following notation in the proof: The matrix direct sum is denoted by $A \oplus B = \text{diag}(A, B)$. The $n \times n$ identity matrix is denoted by \mathbb{I}_n . The Kronecker delta is denoted by δ_{ij} . The components of a matrix $A \in M_n(\mathbb{F}_p)$ are denoted by A_{ij} , with the indices taking values $i, j = 1, \dots, n$. Likewise, the components of a vector $v \in \mathbb{F}_p^n$ are denoted by v_i , with $i = 1, \dots, n$. The inverse a^{-1} is defined to be the multiplicative inverse modulo p of $a \in \mathbb{F}_p$, i.e. the unique $b \in \mathbb{F}_p$ for which $ab \equiv 1 \pmod{p}$.

We first describe a subroutine, termed Algorithm 1, that we will need to call repeatedly.

Proof of correctness: We shall prove that Algorithm 1 works as described. If $A = 0$, then $(P, a, B) = (I, 0, 0)$, which satisfies Eq. (A1), as required. Hence, for the rest of the proof, we shall assume that $A \neq 0$.

We first claim that $c^T A c = a$. Indeed, in the YES case in Line 5, $c^T A c = \sum_{kl} A_{kl} \delta_{ik} \delta_{il} = A_{ii} = a$, and in the NO case in Line 8, we have $A_{ii} = 0$ for all i . Hence, $c^T A c = \sum_{kl} A_{kl} (\delta_{ik} + \delta_{jk}) (\delta_{il} + \delta_{jl}) = A_{ij} + A_{ji} = 2A_{ij} = a$. Note that in both cases, $a \neq 0$. Hence a^{-1} exists.

Next, consider the quadratic form f corresponding to the matrix A :

$$f(t_1, \dots, t_n) = \sum_{ij} A_{ij} t_i t_j = t^T A t. \quad (\text{A7})$$

Define

$$f'(y_1, \dots, y_n) = f(Cy) = \sum_{ij} A_{ij} (Cy)_i (Cy)_j. \quad (\text{A8})$$

By expanding Eq. (A8), and using Eq. (A2), Eq. (A3) and Eq. (A4), we obtain

$$f'(y_1, \dots, y_n) = a \left(y_1 + a^{-1} \sum_{l>1} b_l y_l \right)^2 + g(y_2, \dots, y_n). \quad (\text{A9})$$

Consider the matrix D defined in Eq. (A5). It is easy to see that its inverse has components given by

$$D_{ij}^{-1} = \begin{cases} b_j a^{-1} & i = 1, j \neq 1 \\ \delta_{i,j} & \text{otherwise.} \end{cases} \quad (\text{A10})$$

Let $x = D^{-1}y$. Then $x_1 = y_1 + a^{-1} \sum_{l>1} b_l y_l$ and $x_i = y_i$, for all $i > 1$. Hence,

$$f'(y_1, \dots, y_n) = a x_1^2 + g(x_2, \dots, x_n). \quad (\text{A11})$$

Now the LHS of Eq. (A11) is equal to

$$f'(y) = f'(Dx) = f(CDx) = f(Px) = x^T P^T A P x, \quad (\text{A12})$$

Algorithm 1 Subroutine for Algorithm T

Input: a symmetric matrix $A \in M_n(\mathbb{F}_p)$, where $n \in \mathbb{Z}^+$.

Output: a 3-tuple (P, a, B) , where $P \in \text{GL}_n(\mathbb{F}_p)$, $a \in \mathbb{F}_p$ and $B \in M_{n-1}(\mathbb{F}_p)$ is a symmetric matrix, such that

$$P^T AP = a \oplus B. \quad (\text{A1})$$

- 1: **if** $A = 0$ **then**
- 2: output $(\mathbb{I}_n, 0, 0)$.
- 3: **else** let $a \in \mathbb{F}_p \setminus \{0\}$ and $c \in \mathbb{F}_p^n \setminus \{0\}$ be defined as follows:
- 4: Check if there exists i such that $A_{ii} \neq 0$.
- 5: **if** YES **then**
- 6: Let $I = \min\{i \mid A_{ii} \neq 0\}$.
- 7: Set $a = A_{II}$, $c_j = \delta_{Ij}$ for $j = 1, \dots, n$.
- 8: **else** NO
- 9: Let $(I, J) = \min\{(i, j) \mid A_{ij} \neq 0\}$. (where the minimum is taken with respect to some lexicographic ordering)
- 10: Set $a = 2A_{IJ}$, $c_k = \delta_{Ik} + \delta_{Jk}$ for $k = 1, \dots, n$.
- 11: Choose M for which $c_M \neq 0$.
- 12: Construct the nonsingular matrix $C \in \text{GL}_n(\mathbb{F}_p)$ defined by:

$$C_{ij} = \begin{cases} c_i & j = 1 \\ \delta_{i+1,j} & j \neq 1, i < M \\ 0 & j \neq 1, i = M \\ \delta_{i,j} & j \neq 1, i > M. \end{cases} \quad (\text{A2})$$

for $i, j = 1, \dots, n$.

- 13: Compute

$$b_l = \sum_{ij} c_i A_{ij} C_{jl}, \quad (\text{A3})$$

for $l = 1, \dots, n$.

- 14: Define

$$g(y_2, \dots, y_n) = \sum_{k>1, l>1} \left(\sum_{ij} A_{ij} C_{ik} C_{jl} \right) y_k y_l - a^{-1} \left(\sum_{i>1} b_i y_i \right)^2. \quad (\text{A4})$$

- 15: Construct the matrix $D \in \text{GL}_n(\mathbb{F}_p)$ defined by

$$D_{ij} = \begin{cases} -b_j a^{-1} & i = 1, j \neq 1 \\ \delta_{i,j} & \text{otherwise.} \end{cases} \quad (\text{A5})$$

- 16: Compute the coefficient matrix $B \in M_{n-1}(\mathbb{F}_p)$ of the quadratic form g using

$$B_{ij} = 2^{-1} [g(e_i + e_j) - g(e_i) - g(e_j)], \quad (\text{A6})$$

where e_i is the i th unit vector.

- 17: Compute $P = CD \in \text{GL}_n(\mathbb{F}_p)$.
 - 18: Output (P, a, B) .
-

and the RHS of Eq. (A11) is equal to

$$ax_1^2 + g(x_2, \dots, x_n) = ax_1^2 + \sum_{i,j=1}^n B_{ij} x_i x_j = x^T (a \oplus B)x, \quad (\text{A13})$$

where we used the fact that B is the coefficient matrix of g .

Equating Eq. (A12) and Eq. (A13) then gives $x^T P^T AP x = x^T (a \oplus B)x$. Since this holds for all x , we obtain

$$P^T AP = a \oplus B. \quad (\text{A14})$$

□

We are now ready to describe the algorithm T :

Proof of correctness: If $\alpha = 1$, then Θ is already diagonal. Hence, setting $L = \mathbb{I}$ to be the identity matrix gives the diagonal matrix $L^T \Theta L = \Theta$. Otherwise, we note that for the FOR loop with variable k in Line 4 of Algorithm 2,

Algorithm 2 Algorithm T for matrix diagonalization

Input: a symmetric matrix $\Theta \in M_\alpha(\mathbb{F}_p)$, where $\alpha \geq 1$.

Output: an invertible matrix $L \in GL_\alpha(\mathbb{F}_p)$ such that $L^T \Theta L$ is diagonal.

```

1: if  $\alpha = 1$  then
2:   Output  $L = \mathbb{I}_1 \in M_1(\mathbb{F}_p)$ .
3: else Set  $\Theta_\alpha = \Theta$ .
4:   for  $k = \alpha, \alpha - 1, \dots, 2$  do
5:     Run Algorithm 1 on  $\Theta_k$  to get output  $(P_k, a_k, \Theta_{k-1})$ .
6:   for  $s = 1, \dots, \alpha$  do
7:     Set  $\tilde{P}_s = \mathbb{I}_{\alpha-s} \oplus P_s$ .
8:   Output  $L = \tilde{P}_\alpha \tilde{P}_{\alpha-1} \dots \tilde{P}_2$ .

```

the output of Algorithm 1 on Θ_k is the 3-tuple (P_k, a_k, Θ_{k-1}) that satisfies

$$P_k^T \Theta_k P_k = a_k \oplus \Theta_{k-1}. \quad (\text{A15})$$

Now it is straightforward to show by induction that

$$\tilde{P}_{\alpha-s}^T \dots \tilde{P}_{\alpha-1}^T \tilde{P}_\alpha^T \Theta_\alpha \tilde{P}_\alpha \tilde{P}_{\alpha-1} \dots \tilde{P}_{\alpha-s} = a_\alpha \oplus a_{\alpha-1} \oplus \dots \oplus a_{\alpha-s} \oplus \Theta_{\alpha-s-1}, \quad (\text{A16})$$

for all $s = 0, 1, \dots, \alpha - 2$.

Hence, by using $s = \alpha - 2$ and $\Theta_\alpha = \Theta$, we get

$$\tilde{P}_2^T \dots \tilde{P}_{\alpha-1}^T \tilde{P}_\alpha^T \Theta \tilde{P}_\alpha \tilde{P}_{\alpha-1} \dots \tilde{P}_2 = a_\alpha \oplus a_{\alpha-1} \oplus \dots \oplus a_2 \oplus \Theta_1. \quad (\text{A17})$$

Since each \tilde{P}_s is invertible, their product $L = \tilde{P}_\alpha \tilde{P}_{\alpha-1} \dots \tilde{P}_2$ is also invertible. Therefore, writing $a_1 = \Theta_1 \in \mathbb{F}_p$, we get that

$$L^T \Theta L = \text{diag}(a_\alpha, \dots, a_2, a_1) \quad (\text{A18})$$

is a diagonal matrix.

It is straightforward to see that Algorithm 2 runs in polynomial time in the size of the matrix α .