**QALCO: Quantum Algorithms and Complexity** **2025 - 2026**

Lecture 7: Quantum advantage, RCS, stabilizer states

Lecturer: Alex Bredariol Grilo (alex.bredariol-grilo@lip6.fr)

# 1 Quantum Advantage

The belief that $\mathsf{BPP} \subsetneq \mathsf{BQP}$ tells us that *full-scalable fault-tolerant quantum computers* are more powerful than classical ones. But the big question now is: can we build them? Once we will able to do so, we will be able to solve problems that we cannot solve currently, but an open question is if we can have a task that can be solved by current quantum devices, and which cannot be executed classically.

More concretely, we want to find a task such that:

1. It can be solved efficiently with a near-term quantum experiment.
2. We have some evidence that the problem should be classically hard (no polynomial-time classical solution).
3. The solution should be verifiable with minimal trust in the experiment.
4. The task is useful.

We currently do not have a candidate that satisfies the first three properties, let alone the fourth one. So we will focus on candidate tasks that satisfy the first two properties. In this case, there are two candidate solutions for the task:

- Random Circuit Sampling (RCS)
- Boson Sampling

In this class, we will focus in the first one, but many of the things that we will talk here (specially the classical hardness) also hold for Boson sampling. The choice of talking about RCS is that it uses objects that we already know, instead of having to define bosonic computation.

We are going to define RCS later, and we will see that, almost by definition, it can be solved by simple quantum devices. Moreover, there are implementations of quantum devices which claim of quantum, notably through experiments like Google's "Sycamore" (2019), USTC's "Jiuzhang" (2020), and Google's second experiment (2023). There are many reasons for which we should take such implementations with a grain of salt:

- The device is noisy, while classical hardness results tolerate little noise
- The experiments have a finite size, and classical hardness tells us about asymptotical hardness
- The verification of the experiments take exponential time
- ...

However, even with all of these remarks, their implementation is a landmark of quantum technologies and push forward the quest of quantum advantage.[1]

[1] These lecture notes are based on [2]

### 1.1 Random Circuit Sampling (RCS)

Differently from previous classes, here we will study sampling problems. In a sampling problem, we have an implicit description of a distribution $D$, and the goal is to implement a device whose output distribution is $D$ (or is close to it).

**Definition 1.** *The Random Circuit Sampling problem (RCS) is defined as follows:*

1. *We generate the skeleton of a quantum circuit on $n$ qubits with $d$ layers of random nearest-neighbor gates.*
2. *We fill each gate of this circuit with a Haar random unitary on two qubits.* [2] *We denote this circuit as $C$.*
3. ***Goal:** Sample from the output distribution $D_C$ where the string $x \in \{0,1\}^n$ is picked with probability $|\langle x| C |0^n\rangle|^2$.*

Notice that if we can prove the classical hardness of this problem for a constant $d$, then such problem can be easily solved by (perfect) constant-depth quantum circuits. While this is already challenging, it is much easier than implementing full-scalable quantum computers.

Our goal now is to study the classical hardness of this problem.

### 1.2 More classical complexity classes

In order to study the classical hardness of RCS, we need to see/review other complexity classes.

#### 1.2.1 The polyomial hierarchy

Recall that a language $L$ is in $\mathsf{NP}$ if there exists a polynomial-time predicate $R(x,y)$ and a polynomial $p$ such that

$$x \in L \iff \exists y \in \{0,1\}^{p(|x|)}\ R(x,y).$$

Thus, $\mathsf{NP}$ consists of problems whose membership can be certified by a short proof (or witness) verifiable in polynomial time.

Dually, a language $L$ is in $\mathsf{coNP}$ if

$$x \in L \iff \forall y \in \{0,1\}^{p(|x|)}\ R(x,y),$$

for some polynomial-time predicate $R$. Equivalently, $L \in \mathsf{coNP}$ if its complement is in $\mathsf{NP}$.

The key observation is that $\mathsf{NP}$ corresponds to *one existential quantifier*, while $\mathsf{coNP}$ corresponds to *one universal quantifier* over polynomially bounded strings. This suggests a natural generalization: what happens if we allow *multiple alternating quantifiers*?

More concretely, suppose we allow two alternating quantifiers. For example,

$$x \in L \iff \exists y_1 \forall y_2\ R(x,y_1,y_2),$$

where $R$ is polynomial-time computable.

Intuitively:

- The first player provides a candidate witness $y_1$.
- The second player attempts to refute it by choosing $y_2$.
- The predicate $R$ checks whether the first player survives all possible refutations.

[2] The Haar distribution is the "uniformly random" distribution for continuous distributions.

This leads to the definition of the second level of the hierarchy.

**Definition 2.** *A language $L$ is in $\Sigma_2^P$ if there exist a polynomial $p$ and a polynomial-time predicate $R$ such that*

$$x \in L \iff \exists y_1 \in \{0,1\}^{p(|x|)} \forall y_2 \in \{0,1\}^{p(|x|)}\ R(x, y_1, y_2).$$

Similarly, we define

**Definition 3.** *A language $L$ is in $\Pi_2^P$ if there exist a polynomial $p$ and a polynomial-time predicate $R$ such that*

$$x \in L \iff \forall y_1 \in \{0,1\}^{p(|x|)} \exists y_2 \in \{0,1\}^{p(|x|)}\ R(x, y_1, y_2).$$

We now generalize to any number of alternating quantifiers.

**Definition 4** (The Polynomial Hierarchy)**.** *For $k \geq 1$, a language $L$ is in $\Sigma_k^P$ if there exist a polynomial $p$ and a polynomial-time predicate $R$ such that*

$$x \in L \iff \exists y_1 \forall y_2 \exists y_3 \cdots Q_k y_k\ R(x, y_1, \ldots, y_k),$$

*where:*

- *each $y_i \in \{0,1\}^{p(|x|)}$,*
- *the quantifiers alternate,*
- *and the first quantifier is existential.*

*Similarly, $L$ is in $\Pi_k^P$ if the first quantifier is universal and the quantifiers alternate thereafter.*

Note that:

$$\Sigma_1^P = \mathsf{NP}, \qquad \Pi_1^P = \mathsf{coNP}.$$

**Definition 5.** *The* Polynomial Hierarchy *is*

$$\mathsf{PH} = \bigcup_{k \geq 0} \Sigma_k^P,$$

*where $\Sigma_0^P = \Pi_0^P = \mathsf{P}$.*

**Lemma 1.** *For every $k \geq 0$, the following inclusions hold:*

$$\Sigma_k^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P \qquad \textit{and} \qquad \Pi_k^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P$$

**Theorem 1** (Collapse of the Polynomial Hierarchy)**.** *Let $k \geq 1$. If*

$$\Sigma_k^P = \Pi_k^P,$$

*then the polynomial hierarchy collapses to its $k$-th level, that is,*

$$\mathsf{PH} = \Sigma_k^P = \Pi_k^P.$$

It is believed that $\mathsf{PH}$ does not collapse, and as we will see later, this is the base assumption that will support the quantum advantage on solving RCS.

### 1.2.2 #P

The last piece that we need is the counting version of NP.

**Definition 6** (#P)**.** *A function $f : \{0,1\}^* \to \mathbb{N}$ is in the class #P if there exist a polynomial $p$ and a deterministic polynomial-time Turing machine $M$ such that for every input $x$,*

$$f(x) = \#\{\, w \in \{0,1\}^{p(|x|)} \;:\; M(x,w) \text{ accepts}\,\}.$$

*In other words, $f(x)$ counts the number of accepting witnesses of $M$ on input $x$.*

**Theorem 2** (Toda's theorem)**.** $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$ .

Notice the following corollary from Toda's theorem.

**Corollary 1.** *Let $A$ be a #P-hard problem. If there is some $k \in \mathbb{N}$ such that $A \in \Sigma_k^P$, then $\mathsf{PH} = \Sigma_k^P = \Pi_k^P$.*

## 1.3 Classical sums and the Stockmeyer's algorithm

In this section, we will discuss a problem that we will call "classical sum".

**Definition 7** (Classical sum)**.** *Given a classical circuit computing $f : \{0,1\}^n \to \{0,1\}$, compute the sum $\sum_{x\in\{0,1\}^n} f(x)$.*

**Theorem 3.** *The classical sum problem is #P-hard.*

**Definition 8** (Approximate classical sum)**.** *Given the same input as the classical sum problem, output a multiplicative estimate $\alpha$ such that:*

$$(1-\epsilon)\sum_{x\in\{0,1\}^n} f(x) \le \alpha \le (1+\epsilon)\sum_{x\in\{0,1\}^n} f(x)$$

We will see now that approximating classical sums is much easier than computing them exactly.

**Theorem 4** (Stockmeyer)**.** *There exists a probabilistic polynomial-time algorithm with access to an NP oracle that, given $x$ and $\varepsilon > 0$, outputs a value $\widetilde{f}(x)$ such that with high probability,*

$$(1-\varepsilon)f(x) \le \widetilde{f}(x) \le (1+\varepsilon)f(x).$$

*In other words, approximating classical sums is in $\mathrm{BPP}^{\mathsf{NP}} \subseteq \Sigma_2^P$.*

*Proof.* Let $S = \{x : f(x) = 1\}$, so $|S| = \sum_x f(x)$. The idea is to estimate $|S|$ using random hashing.

We choose a hash function

$$h_k : \{0,1\}^m \to \{0,1\}^\ell$$

from a pairwise independent family. Such families of functions exist unconditionally, and the property that we need from them is that

$$\Pr_k[h_k(y) = 0^\ell] = 2^{-\ell}.$$

Thus the expected number of solutions mapping to $0^\ell$ is

$$\mathbb{E}_k[|\{y \in S : h(y) = 0^\ell\}|] = |S| \cdot 2^{-\ell}.$$

If $\ell$ is chosen so that $2^\ell \approx |S|$, then with good probability we have that

- If $|S| \gg 2^k$, then with high probability some element hashes to $0^k$.
- If $|S| \ll 2^k$, then with high probability none do.

Therefore, our goal is to check if there exists $x$ such that

$$f(x) = 1 \quad \text{and} \quad h_k(y) = 0^\ell.$$

Notice that this is an NP predicate, so we can use an NP oracle to check if this is true. So the algorithm for estimating $|S|$ would be the following:

1. For $\ell = 0, 1, \ldots, m$:
   (a) Pick hash functions $h_k$.
   (b) Use the NP oracle to check whether

$$\exists x \text{ s.t. } f(x) = 1 \quad \text{and} \quad h_k(y) = 0^\ell.$$

2. Estimate the largest $\ell^*$ for which the above is true with non-negligible probability.
3. Output $2^{\ell^*}$.

Repeating the experiment sufficiently many times and using Chernoff bounds ensures a multiplicative $(1 \pm \varepsilon)$ approximation with high probability. □

One particular consequence of Stockmeyer's algorithm will be a reduction from classical samplers to computing probability distributions:

**Theorem 5.** *Let $D$ be a distribution on $\ell$ bits. If there is a polynomial-time algorithm $S$ such that for every $x \in \{0,1\}^\ell$, $\Pr_{r \in \{0,1\}^p}[S(r) = x] = D(x)$, then for every $y \in \{0,1\}^\ell$ we can approximate $D(y)$ in $\mathsf{BPP}^{\mathsf{NP}} \subseteq \Sigma_2^P$.*

*Proof.* Let us define the function $f_y : \{0,1\}^p \to \{0,1\}$ such that $f_y(r) = \mathbf{1}_{S(r)=y}$. Then we can use Stockmeyer's algorithm to approximate $\frac{1}{2^p} \sum_y f(y) = D(y)$. □

### 1.4 Quantum sums

We will now study a slight variation of the classical sum problem that we will call "quantum sum".

**Definition 9** (Quantum sum problem)**.** *Given a classical circuit computing $g : \{0,1\}^n \to \{\pm 1\}$, compute the sum $\sum_{x \in \{0,1\}^n} g(x)$.*

**Theorem 6.** *The quantum sum problem is also #P-hard.*

As in the classical case, we can consider the approximate version of the problem.

**Definition 10** (Approximate quantum sum)**.** *Given the same input as the classical sum problem, output a multiplicative estimate $\alpha$ such that:*

$$(1 - \epsilon) \sum_{x \in \{0,1\}^n} g(x) \leq \alpha \leq (1 + \epsilon) \sum_{x \in \{0,1\}^n} g(x)$$

However, differently from the classical case, this problem remains #P-hard in its approximation version.

**Theorem 7.** *The approximate quantum sum problem is also #P-hard.*

The intuition why this problem remains hard is that exponential-size cancellations (interference) make this problem much harder than the classical approximate sum.

**Exercise 1.** *Given that approximate quantum sum is #P-hard, argue that computing sign*$(\sum_{x\in\{0,1\}^n} g(x))$ *is also #P-hard.*

**Exercise 2.** *Given that computing sign*$(\sum_{x\in\{0,1\}^n} g(x))$ *is #P-hard, argue that approximating* $(\sum_x g(x))^2$ *is also #P-hard.*

We will now connect such quantum sums and quantum circuits. For that, we will consider the circuit in Figure 1, where we consider the oracle $O_g |x\rangle \mapsto (-1)^{g(x)} |x\rangle$, for $x \in \{0,1\}^\ell$.

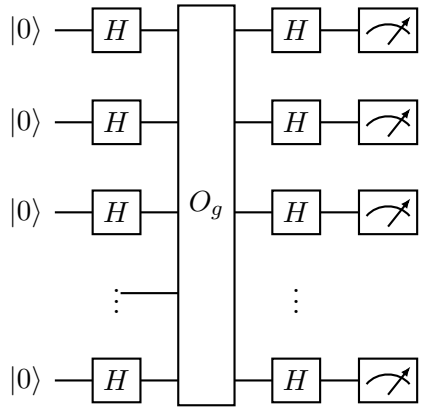


Figure 1: Quantum circuit $\tilde{C}_g$.

**Exercise 3.** *The probability that the circuit outputs* $0^\ell$ *is* $\frac{1}{2^\ell}(\sum_x g(x))^2$.

**Theorem 8.** *Let* $D_{C_g}$ *be the distribution induced by the measurement of the output of* $C_g |0^\ell\rangle$. *Suppose that there is a classical algorithm that can sample from* $D_{C_g}$. *Then the polynomial hierarchy collapses.*

*Proof.* If such a classical sampler exists, then from Theorem 5, it means that we can compute $D_{C_g}(0^\ell)$ in $\Sigma_2^P$. But computing such a value is #P-hard (Exercise 2). So, from Corollary 1, the polynomial hierarchy collapses. □

### 1.5 Not the end of the story

**RCS.** We said that we wanted to show that sampling from a random circuit is hard, and here we only showed the hardness for a very specific #P-hard circuit. To go further, we need a way to do worst-to-average case reductions.

We won't get into details here, but one way of doing it is with the so-called Caley paths.

This technique gives a way of interpolating between quantum circuits. In particular one can use the Cayley parametrization to interpolate between a fixed circuit and a completely random circuit with the same architecture: Given a real parameter $0 \leq \theta \leq 1$, one defines a family of circuits $C(\theta)$ such that

1. $C(0)$ is the circuit of interest $C$
2. $C(1)$ is a circuit with the same architecture as $C$ whose gates instantiate unitaries drawn independently from the Haar measure. Moreover, it can be shown that the statistical distance of $C(1-\delta)$ and circuits composed of Haar random gates is $\Theta(\Delta)$, for small $|\delta| \leq \Delta$.

A nice feature of the Cayley parametrization is that it tolerates some noise in the reconstruction of the output probability of interest. For example, given $(\theta_1, y_1), ..., (\theta_\ell, y_\ell)$ such that $\left|y_i - ||\Pi C(\theta_i)\,|\psi\rangle\,||^2\right| \leq \epsilon$, we can compute a rational function $f$ such that

$$\left|f(0) - ||\Pi C\,|\psi\rangle\,||^2\right| \leq \eta, \tag{1}$$

where $\eta$ depends on the degree of the extrapolation, $\epsilon$, and the extrapolation point $\theta = 0$.

You can find more details in Appendix A.

**Exact sampling.** We only discussed the impossibility of an efficient classical sampler $S$ that samples from the same distribution as $C$. We can work it harder to show hardness of approximating such a distribution.

**Noisy circuits.** Even if we talk about approximation, we are still talking about the distribution coming from ideal circuits. However, we don't expect to be able to sample such distributions even with very good quantum devices, due to noise. So, we need to understand if the classical hardness also holds for the distributions of noisy RCS. Actually, this question is much more subtle: we know that if the circuit from RCS is noisy enough, then there is a classical algorithm that can approximate the output distribution [1]. So one needs to figure out up to which value of noise the problem becomes easy, and from which value we have indication that it is classically hard.

# 2 Stabilizer States and the Gottesman–Knill Theorem

## 2.1 The Pauli Group

We begin with the single-qubit Pauli matrices:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The *single-qubit Pauli group* is

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}.$$

For $n$ qubits, the Pauli group is

$$\mathcal{P}_n = \{\omega P_1 \otimes \cdots \otimes P_n : \omega \in \{\pm 1, \pm i\},\ P_j \in \{I, X, Y, Z\}\}.$$

Pauli operators either commute or anticommute:

$$PQ = \pm QP.$$

## 2.2 Stabilizer States

**Definition 11.** *A* stabilizer group *on $n$ qubits is an abelian subgroup $S \subseteq \mathcal{P}_n$ such that:*

- *$-I \notin S$,*
- *$S$ is generated by $n$ independent commuting Pauli operators.*

**Definition 12.** *A quantum state $|\psi\rangle$ is a* stabilizer state *if there exists a stabilizer group $S$ such that*

$$|\psi\rangle$$

*is the unique state satisfying*

$$P|\psi\rangle = |\psi\rangle \quad \textit{for all } P \in S.$$

### 2.3 Examples

**Example 1: The $|0\rangle$ State.** The state $|0\rangle$ is stabilized by $Z$, since

$$Z\,|0\rangle = |0\rangle\,.$$

**Example 2: The Bell State.** Consider

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

It is stabilized by:

$$X \otimes X, \qquad Z \otimes Z.$$

### 2.4 Clifford Group

**Definition 13.** *The* Clifford group $\mathcal{C}_n$ *is the set of unitary operators $U$ such that*

$$U\mathcal{P}_n U^\dagger = \mathcal{P}_n.$$

*That is, Clifford operators map Pauli operators to Pauli operators under conjugation.*

Important examples:

- Hadamard gate:
$$HXH = Z, \qquad HZH = X.$$
- Phase gate:
$$SXS^\dagger = Y, \qquad SZS^\dagger = Z.$$
- CNOT gate:
$$\mathrm{CNOT}(X \otimes I)\mathrm{CNOT} = X \otimes X,$$
$$\mathrm{CNOT}(I \otimes Z)\mathrm{CNOT} = Z \otimes Z.$$

These gates generate the Clifford group.

### 2.5 The Gottesman–Knill Theorem

**Theorem 9** (Gottesman–Knill)**.** *Any quantum computation that:*

- *starts in the state $|0^n\rangle$,*
- *applies only Clifford gates,*
- *performs a measurements in the first qubit in the computational basis,*

*can be simulated efficiently on a classical computer.*

*Proof.* To simulate this computation, we start with the stabilizer of the all-zeroes state: $S_0 = \{Z \otimes I \otimes ... \otimes I, I \otimes Z \otimes .... \otimes I, ..., I \otimes I \otimes ... \otimes Z\}$.

We can show that after each Clifford gate, we can efficiently compute the stabilizers of the state step of the computation. For that, let $S_i$ be the stabilizer set of the state after the application of the $i$-th Clifford gate and let $U$ be the $(i+1)$-the gate.

**Exercise 4.** *Show that*

$$U|\psi\rangle \text{ is stabilized by } USU^\dagger.$$

This allows to compute the stabilizers $S_m$ after the last Clifford gate. We describe how to simulate a measurement of the observable $Z_1$ in polynomial time.

For any Pauli observable $P$, exactly one of the following holds:

1. $P$ commutes with every generator of $\mathcal{S}$.
2. $P$ anticommutes with at least one generator.

The behavior of the measurement depends entirely on this distinction.

### 2.6 Case 1

Suppose $Z_j$ commutes with every stabilizer generator:

$$[Z_j, S_i] = 0 \quad \text{for all } i.$$

Then $Z_j$ belongs to the normalizer of $\mathcal{S}$ and the state $|\psi\rangle$ is already an eigenstate of $Z_j$. Moreover, there exists a product of generators equal (up to phase) to $Z_j$:

$$Z_j = \pm \prod_{i \in I} S_i.$$

The measurement outcome is:

$$+1 \quad \text{if the phase is } +1,$$

$$-1 \quad \text{if the phase is } -1,$$

and we can compute which case we have with Gaussian elimination.

Notice that in this case, the stabilizer group remains unchanged since the measurement does not disturb the state.

### 2.7 Case 2

Suppose $Z_j$ anticommutes with at least one generator.

**Exercise 5.** *Show that the measurement outcome is uniformly random in* $\{\pm 1\}$.

For updating the stabilizer of the state after the measurement, we have:

**Step 1: Choose a pivot generator.** Pick a generator $S_k$ such that

$$\{Z_j, S_k\} = 0.$$

**Step 2: Make other generators commute.** For every generator $S_\ell$ that anticommutes with $Z_j$, replace

$$S_\ell \leftarrow S_\ell S_k.$$

After this step, all generators except $S_k$ commute with $Z_j$.

**Step 3: Replace the pivot.** Remove $S_k$ and replace it with

$$S'_k = \pm Z_j,$$

where the sign is chosen according to the measurement outcome.

**State update.** The new stabilizer group has generators

$$\{S_1, \ldots, S_{k-1}, S'_k, S_{k+1}, \ldots, S_n\}.$$

The post-measurement state is stabilized by this new group. □

Notice that we did not really used the fact that we started in the all-zeroes state and that we measured the first qubit the the $Z$-basis. Indeed, the algorithm can be generalized and we can start with any stabilizer state and measure with any Pauli observable.

## References

[1] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 945–957, New York, NY, USA, 2023. Association for Computing Machinery.

[2] Bill Fefferman. On the theory of near-term quantum advantage. `https://www.ias.edu/sites/default/files/billfefferman-lectures-pcmi-out.pdf`.

## A Caley paths

The Cayley path is a parameterized path that interpolates between two unitaries [**?**]. Namely, let $\mathbb{U}(N)$ be the set of $N \times N$ unitary matrices and $U_0, U_1 \in \mathbb{U}(N)$. Then, the Cayley path is a parameterized path $U(\theta)$ such that $U(\theta) \in \mathbb{U}(N)$ for all $\theta \in [0, 1]$ and $U(0) = U_0$ and $U(1) = U_1$ based on the Cayley function

$$f(x) = \frac{1 + ix}{1 - ix}, \tag{2}$$

where we define $f(-\infty) = -1$. Here, we (informally) discuss the Cayley path for interpolating between a quantum circuit $C$ and a Haar random circuit following the presentation in [**?**].

Let $C = \mathcal{C}_m \cdots \mathcal{C}_1$ be a quantum circuit acting on $n$ qubits with $m$ local unitary gates. Suppose that $\mathcal{C}_k = C_k \otimes I$ so that $C_k$ is a one or two-qubit gate with identity on the rest of the qubits. Given a local gate $H_k$, then there exists a unique Hermitian matrix $h_k$ such that $H_k = f(h_k)$. Suppose that $H_k$ is Haar random gate that is the same size as a gate $C_k$ in the quantum circuit. Then, the Cayley path for each gate is given by

$$C_k(\theta) = C_k f(\theta h_k), \tag{3}$$

where $H_k = f(h_k)$, thus $C_k(0) = C_k$ and $C_k(1) = C_k H_k$ is a Haar random gate. The full quantum circuit has the Cayley path $C(\theta) = \mathcal{C}_m(\theta) \cdots \mathcal{C}_1(\theta)$, where $\mathcal{C}_k(\theta) = C_k(\theta) \otimes I$.

Crucially, [**?**] showed that if one runs $C(\theta_i)$ for different choices of $\theta_i$ and estimates a measurement $p(\theta_i)$ probability for a certain bitstring, then one can recover a function $p(\theta)$ that gives the measurement probability for $C(\theta)$ for any choice of $\theta$ as long as the algorithm is given enough points with the necessary precision. Recall from the construction of the Cayley path [**?**] that the probabilities are rational function of type $(\Theta(m), \Theta(m))$ with factorizable denominators whose determination can be reduced to polynomial extrapolation [**?**]. The recovery can be done via Lagrange extrapolation as shown in Eq. 13 in [**?**]. Below we denote by $e(\theta)$ the error probability which is the difference of the exact and estimated probabilities obtained from Lagrange extrapolation or generalized Berlekamp-Welch.

**Lemma 2.** *Suppose $e(\theta)$ is a degree $d$ polynomial in $\theta$. Assume $e(\theta_i) \leq \epsilon$ where $|1 - \theta_i| \in [0, \Delta]$. Then $|e(0)| \leq \epsilon \frac{\exp[d(1+\log \Delta^{-1})]}{\sqrt{2\pi d}}$.*

Let $p_{C(1)}$ be the probability distribution over circuits with Haar local unitary gates and $p_{C(\theta)}$ the probability distribution over circuits whose gates are deformed according to Eq. (3) then

**Lemma 3.** *The total variation distance* $TVD\left(p_{C(\theta)}, p_{C(1)}\right) = O(m\Delta)$, *for* $|1-\theta| \leq \Delta$.