

# Approximating Output Probabilities of Shallow Quantum Circuits which are Geometrically-local in any Fixed Dimension.

Suchetan Dontha, Shi Jie Samuel Tan, Stephen Smith,  
Sangheon Choi, Matthew Coudron

## Abstract

We present a classical algorithm that, for any  $D$ -dimensional geometrically-local, quantum circuit  $C$  of polylogarithmic-depth, and any bit string  $x \in \{0, 1\}^n$ , can compute the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within any inverse-polynomial additive error in quasi-polynomial time, for any fixed dimension  $D$ . This is an extension of the result [CC21], which originally proved this result for  $D = 3$ . To see why this is interesting, note that, while the  $D = 1$  case of this result follows from a standard use of Matrix Product States, known for decades, the  $D = 2$  case required novel and interesting techniques introduced in [BGM20]. Extending to the case  $D = 3$  was even more laborious, and required further new techniques introduced in [CC21]. Our work here shows that, while handling each new dimension has historically required a new insight, and fixed algorithmic primitive, based on known techniques for  $D \leq 3$ , can now handle any fixed dimension  $D > 3$ .

Our algorithm uses the Divide-and-Conquer framework of [CC21] to approximate the desired quantity via several instantiations of the same problem type, each involving  $D$ -dimensional circuits on about half the number of qubits as the original. This division step is then applied recursively, until the width of the recursively decomposed circuits in the  $D^{\text{th}}$  dimension is so small that they can effectively be regarded as  $(D - 1)$ -dimensional problems by absorbing the small width in the  $D^{\text{th}}$  dimension into the qudit structure at the cost of a moderate increase in runtime. The main technical challenge lies in ensuring that the more involved portions of the recursive circuit decomposition and error analysis from [CC21] still hold in higher dimensions, which requires small modifications to the analysis in some places. Our work also includes some simplifications, corrections and clarifications of the use of block-encodings within the original classical algorithm in [CC21].

## 1 Introduction

It is known that it is  $\#P$ -hard to compute the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within  $2^{-n^2}$  additive error for low-depth, geometrically-local quantum circuits  $C$  [Mov20, KMM21], and worst-case hardness results for this task date back to [TD04]. However, these hardness results indicate that computing output probabilities with such small additive error is almost certainly out of reach for both classical *and* quantum computers. If we restrict our attention to additive errors that are achievable with quantum computers, such as inverse polynomial additive error achievable by taking polynomially many samples from the quantum circuit  $C$ , then classical hardness for this estimation problem is much less clear. In fact, [BGM20] introduced an elegant classical polynomial time algorithm for this estimation task in the case of 2D circuits. Their algorithm makes a novel use of 1D Matrix Product States carefully tailored to the 2D geometry of the circuit in question. While it is

not clear how to generalize the techniques of [BGM20] to higher dimensional circuits, [CC21] introduced a Divide-and-Conquer algorithm that can compute the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within any inverse-polynomial additive error in quasi-polynomial time for any  $3D$ , constant-depth quantum circuit  $C$ . The algorithm in [CC21] works by recursively subdividing the quantum circuit  $C$  into pieces, constructed using block-encodings, and introduces new techniques for analyzing the extent to which quantum entanglement between different qubits can impact the global quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$ .

Given the progression of ideas required to classically approximate the output probabilities of higher dimensional quantum circuits, it is natural to wonder what would be required to go even further. In this work we will show that there exists a classical quasi-polynomial time algorithm which can compute  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to inverse polynomial additive error for any constant-depth, geometrically-local quantum circuits of any fixed dimension  $D$ .

**Theorem 1 (Main Result).** *For any  $D$ -dimensional geometrically-local, constant depth quantum circuit  $C$  acting on  $n$  qubits, the algorithm  $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$  computes the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within  $\delta$  additive error in time  $\delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D \cdot 3^D})}$ .<sup>1</sup>*

Our main result illustrates that our toolkit for classically simulating low-depth geometrically local quantum circuits no longer requires a conceptual leap to advance from  $D$ -dimensional circuits to  $(D + 1)$ -dimensional circuits. At least not when the runtime of our classical algorithm can be quasi-polynomial. Rather, one fixed divide-and-conquer algorithmic primitive allows us to inductively establish an additive-error classical simulation algorithm for any dimension  $D$ .

The rest of this paper will proceed as follows. In section 2 we will review block-encodings and syntheses, both of which are used extensively throughout the algorithm. Note that section 2 primarily consists of definitions and lemmas from [CC21] that are tweaked for clarity and correctness. In section 3 we will provide the pseudocode for our algorithm and prove our main result. Following this we will provide the runtime and error analysis for our algorithm in sections 3.1 and 3.2 respectively.

## 2 Block-encodings and Syntheses

In order to state the pseudocode for our algorithms in Section 3 below we first need to establish a way to construct the “recursive subdivisions” of the quantum circuit  $C$  that our divide-and-conquer algorithm iteratively creates. We will concretely describe these subdivisions as “syntheses”, as defined in [CC21] and reviewed here for the convenience of the reader. Syntheses themselves use the idea of a block-encoding which we paraphrase below from [GSLW19].

**In order to understand the following discussion, which is essential to the rest of this paper, it is necessary to read sections 2, 3, and 4 of [CC21]. The lemmas repeated in this section are only included here in order to clarify or correct certain definitions in section 3 of [CC21]. Many other definitions and lemmas from sections 2, 3, and 4 of [CC21] are not repeated here and must be read from the original document (see the arxiv version at <https://arxiv.org/pdf/2012.05460.pdf>).**

**Definition 2 (Block-encoding).** Suppose that  $A$  is an  $s$ -qubit operator,  $\alpha, \epsilon \in \mathbb{R}_+$  and  $a \in \mathbb{N}$ . Then

---

<sup>1</sup>Here  $S = (C, L, M, N)$  is the synthesis describing circuit  $C$ , as defined in this paper and in [CC21], and  $\mathcal{B}$  is our base-case algorithm which we specify to be the 2D algorithm of [BGM20], and which our algorithm uses to solve subproblems which have been recursively subdivided down to 2 dimensions.

we say that the  $(s + a)$ -qubit unitary  $U$  is an  $(\alpha, a, \epsilon)$ -block-encoding of  $A$ , if

$$\left\| A - \alpha(\langle 0|^{\otimes a} \otimes I)U(|0\rangle^{\otimes a} \otimes I) \right\| \leq \epsilon.$$

Consider a cut  $B \cup M \cup F$  made anywhere in the cube and let  $\sigma_{MUF} = \text{tr}_B (C_{BUMUF} |0\rangle\langle 0|_{BUMUF} C_{BUMUF}^\dagger)$ . The following result is obtained by applying Lemma 45 of [GSLW19]:

**Lemma 3** (Block-encoding for  $\sigma_{MUF}$ ). *The following is a  $(1, |B \cup M \cup F|, 0)$ -block-encoding of  $\sigma_{MUF}$ :*

$$\Gamma = \left( C_{BUM'UF'}^\dagger \otimes I_{MUF} \right) (I_B \otimes \text{SWAP}_{MUF, M'UF'}) (C_{BUM'UF'} \otimes I_{MUF}).$$

In the above,  $C_{BUM'UF'}$  is notation to indicate that we will be applying the circuit  $C_{BUMUF}$  on the registers  $B$ ,  $M'$ , and  $F'$ . In other words,

$$\sigma_{MUF} = (\langle 0|_{BUM'UF'} \otimes I_{MUF}) \Gamma (|0\rangle_{BUM'UF'} \otimes I_{MUF}).$$

The registers  $M'$  and  $F'$  above are copies of the registers  $M$  and  $F$ , respectively, and are introduced by Lemma 45 of [GSLW19]. By interleaving  $M'$  with  $M$  and  $B'$  with  $B$  and adding swap gates where appropriate, we can ensure that the resulting circuit,  $\Gamma$ , is still geometrically-local and has depth at most 3 times the depth of  $C_{BUMUF}$ . By simply moving the  $M$  register in Lemma 3 to the set of registers which are post-selected, we see that  $\Gamma$  is also a block-encoding of  $\rho_F := \langle 0|_M \sigma_{MUF} |0\rangle_M$ .

**Lemma 4** (Block-encoding for  $\rho_F$ ). *The block-encoding introduced in Lemma 3,  $\Gamma$ , is a  $(1, |B \cup F| + 2|M|, 0)$ -block-encoding of  $\rho_F$ . Note that Lemma 4 is a correction of Lemma 7 of [CC21].*

*Proof.*

$$\begin{aligned} & (\langle 0|_{BUM'UF'UM} \otimes I_F) \Gamma (|0\rangle_{BUM'UF'UM} \otimes I_F) \\ &= \langle 0|_M (\langle 0|_{BUM'UF'} \otimes I_F) \Gamma (|0\rangle_{BUM'UF'} \otimes I_F) |0\rangle_M \\ &= \langle 0|_M \sigma_{MUF} |0\rangle_M \\ &= \rho_F. \end{aligned}$$

□

Since  $\rho_F$  is the state that we are really interested in, we will henceforth refer to  $\Gamma$  as  $\Gamma_{\rho_F}$ . We can now iteratively apply Lemma 53 from [GSLW19] to obtain a block-encoding for  $\rho_F^k$  for any integer  $k \geq 1$ . To do this, we will need  $k - 1$  copies of each of the registers  $B$ ,  $M'$ ,  $F'$ , and  $M$ . Let  $B_1 = B$ ,  $M'_1 = M'$ ,  $F'_1 = F'$ , and  $M_1 = M$ . Furthermore, for each  $i$ ,  $1 < i \leq k$ , let  $B_i$ ,  $M'_i$ ,  $F'_i$ ,  $M_i$  be copies of  $B$ ,  $M'$ ,  $F'$ , and  $M$ , respectively.

**Lemma 5** (Block-encoding for  $\rho_F^k$ ). *The following is a  $(1, k|B \cup M' \cup F' \cup M|, 0)$ -block-encoding of  $\rho_F^k$ :*

$$\Gamma_{\rho_F^k} = \prod_{i=1}^k \left( C_{B_i M'_i F'_i}^\dagger \otimes I_{M_i U F} \right) \left( I_{B_i} \otimes \text{SWAP}_{M_i U F, M'_i U F'_i} \right) \left( C_{B_i M'_i F'_i} \otimes I_{M_i U F} \right).$$

In other words,

$$\rho_F^k = \left( \langle 0|_{\mathcal{B}_k \cup \mathcal{M}'_k \cup \mathcal{F}'_k \cup \mathcal{M}_k} \otimes I_F \right) \Gamma_{\rho_F^k} \left( |0\rangle_{\mathcal{B}_k \cup \mathcal{M}'_k \cup \mathcal{F}'_k \cup \mathcal{M}_k} \otimes I_F \right)$$

where  $\mathcal{B}_k = B_1 \cup B_2 \cup \dots \cup B_k$ ,  $\mathcal{M}'_k = M'_1 \cup M'_2 \cup \dots \cup M'_k$ , etc. Note that this is a correction of equation 7 of [CC21]

**Lemma 6** (Block-encoding for  $\rho_B^k$ ). *Analogously, the following is a  $(1, k|F \cup M' \cup B' \cup M|, 0)$ -block-encoding of  $\rho_B^k$ :*

$$\Gamma_{\rho_B^k} = \prod_{i=1}^k \left( C_{B'_i \cup M'_i \cup F_i}^\dagger \otimes I_{M_i \cup B} \right) \left( I_{F_i} \otimes \text{SWAP}_{M_i \cup B, M'_i \cup B'_i} \right) \left( C_{B'_i \cup M'_i \cup F_i} \otimes I_{M_i \cup B} \right).$$

In other words,

$$\rho_B^k = \left( \langle 0 |_{\mathcal{F}_k \cup \mathcal{M}'_k \cup \mathcal{B}'_k \cup \mathcal{M}_k} \otimes I_B \right) \Gamma_{\rho_B^k} \left( |0 \rangle_{\mathcal{F}_k \cup \mathcal{M}'_k \cup \mathcal{B}'_k \cup \mathcal{M}_k} \otimes I_B \right)$$

where  $\mathcal{F}_k = F_1 \cup F_2 \cup \dots \cup F_k$ ,  $\mathcal{M}'_k = M'_1 \cup M'_2 \cup \dots \cup M'_k$ , etc.

Note that this is a correction for equation 7 of [CC21]

Importantly, we are free to interleave all of the copies of the registers  $B$ ,  $M$  and  $F$  with their originals. We do this in such a way so that we can minimally pad each 2-qubit gate from  $C_{B \cup M \cup F}$  with swap gates so that this new ‘padded’ circuit is still geometrically local. Furthermore, the depth of this new padded circuit is at most  $(2k + 1)$  times the original depth of  $C$ .

**Definition 7** (Synthesis). We say that an unnormalized quantum state  $\phi$  is *synthesized* by a quantum circuit  $\Gamma$ , if  $\Gamma$  has three registers of qubits  $L, M, N$  such that:

$$\phi = \phi_{(\Gamma, L, M, N)} = \text{tr}_{L \cup M}(\langle 0_M | \Gamma | 0_{L \cup M \cup N} \rangle \langle 0_{L \cup M \cup N} | \Gamma^\dagger | 0_M \rangle). \quad (1)$$

In this case we say that the circuit  $\Gamma$  together with a specification of the registers  $L, M, N$  constitutes a *synthesis* of  $\phi$ . When  $\phi$  is implicit we will call this collection  $(\Gamma, L, M, N)$  a *synthesis*. This definition was taken directly from [CC21] and is only here for the convenience of the reader. All syntheses explicitly used in the rest of this paper are defined in section 4 of [CC21].

### 3 Algorithms and Analysis

Having discussed the essential concepts of Syntheses and block-encodings in Section 2 above, we now give explicit description of our classical simulation algorithm below. Our algorithm is divided into two pieces, Algorithm 1 and Algorithm 2. Algorithm 1 simply handles some technical edge cases for the error parameter  $\delta$ , and then identifies the ‘heavy slices’ of the circuit  $C$ , which sets the stage for making a call to Algorithm 2. Algorithm 2 contains the actual divide-and-conquer structure, describing how to subdivide the circuit using syntheses and then perform recursive calls to Algorithm 1 in order to compute the sub-problem of the divide-and-conquer process which corresponds to each synthesis. The particular

---

**Algorithm 1:**  $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$ : Quasi-Polynomial Time Additive Error Approximation for  $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$ .

---

**Input :** Synthesis  $S = (C, L, M, N)$  where  $C$  is a  $D$ -Dimensional Geometrically-Local,  $\mathcal{B}$  a base case algorithm for 2D circuits, depth- $d$  circuit, approximation error  $\delta$ , dimension  $D$

**Output:** An approximation of  $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$  to within additive error  $\delta$ .

*/\* We begin by handling the case in which  $\delta$  is so small that it trivializes our runtime, and the case in which  $\delta$  is so large that it causes meaningless errors: \*/*

```

1 if  $\delta \leq 1/n^{\log^2(n)}$  then
2   return The value  $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$  computed with zero error by a “brute force”
    $2^{O(n)}$ -time algorithm.
3 if  $\delta \geq 1/2$  then return  $1/2$ 
4 if  $D = 2$  then
5   return  $\mathcal{B}(S, \delta)$ 
/* Here begins the non-trivial part of the algorithm: */
6 Let  $N$  be the register containing all of the qubits on which  $C$  acts. Since these qubits are
   arranged in a hyper-cubic lattice, one of the sides of the hyper-cube  $N$  must have length
   at most  $n^{\frac{1}{D}}$ . We will call the length of this side the “width” and will now describe how to
   “cut” the hyper-cube  $N$ , and the circuit  $C$ , perpendicular to this particular side.
7 Select  $\frac{1}{10d}n^{\frac{1}{D}}$  light-cone separated slices  $K_i$  of  $10d$  width in  $N$ , with at most  $10d$  distance
   between adjacent slices. Let  $h(n) = \log^7(n)$ . Run Algorithm 2 as
    $\mathcal{A}_{full}(S, \mathcal{B}, 2^{\frac{\log(\delta)}{2h(n)}-1} - 2^{\frac{\log(\delta)}{h(n)}-1}, D-1)$  to check if at least  $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$  of the slices obey:
8

```

$$\left| \text{tr} \left( \langle 0_{M_i} | C | 0_{ALL} \rangle \langle 0_{ALL} | C^\dagger | 0_{M_i} \rangle \right) \right| \geq 2^{\frac{\log(\delta)}{h(n)}}.$$

```

9 OR, there are fewer than  $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$  slices that obey:
10

```

$$\left| \text{tr} \left( \langle 0_{M_i} | C | 0_{ALL} \rangle \langle 0_{ALL} | C^\dagger | 0_{M_i} \rangle \right) \right| \geq 2^{\frac{\log(\delta)}{h(n)}}.$$

*/\* See the runtime analysis in the proof of Theorem 28 of [CC21] for a detailed explanation of how the aforementioned run of Algorithm 2 which is the analog for the base case algorithm  $\mathcal{B}$  in [CC21] can efficiently distinguish between the above two cases (via Remark 6 in [CC21]). \*/*

```

11 if Fewer than  $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$  of the slices obey Line 10 then return 0
12 if At least  $\frac{1}{10d}n^{\frac{1}{D}} - h(n)$  of the slices obey Line 10 then
13   We will denote the set of these slices by  $K_{heavy}$ . Note that the maximum amount of
   width between any two adjacent slices in  $K_{heavy}$  is  $10d \cdot h(n)$ . Furthermore, the
   maximum amount of width collectively between  $\Delta$  slices in  $K_{heavy}$  is
    $10d\Delta + 10d \cdot h(n)$ . Now that the set  $K_{heavy}$  has been defined, we will use this fixed set
   in the recursive algorithm, Algorithm 2.
14 return
    $\mathcal{A}(S, \eta = \frac{\log(n)}{D \log(4/3)}, \Delta = \log(n), \epsilon = \delta 2^{-10 \log(n) \log(\log(n))}, h(n) = \log^7(n), K_{heavy}, D, \mathcal{B})$ 

```

---

---

**Algorithm 2:**  $\mathcal{A}(S, \eta, \Delta, \epsilon, h(n), K_{heavy}, D, \mathcal{B})$ : Recursive Divide-and-Conquer Subroutine for Algorithm 1.

---

**Input :** D-dimensional Geometrically-Local, depth- $d$  synthesis  $S$ , number of iterations  $\eta$ , number of cuts  $\Delta$ , positive base-case error bound  $\epsilon > 0$ , a set of heavy slices  $K_{heavy}$ , dimension  $D$ ,  $\mathcal{B}$  a base case algorithm for 2D circuits

**Output:** An approximation of the quantity  $\langle 0_N | \phi_S | 0_N \rangle$  where  $\phi_S$  is the un-normalized mixed state specified by the D-dimensional geometrically-local, depth- $d$  synthesis  $S$ , and  $|0_N\rangle$  is the 0 state on the entire  $N$  register of that synthesis. The approximation error is bounded in the analysis below.

- 1 Given the geometrically-local, depth- $d$  synthesis  $S = (\Gamma, L, M, N)$ , let us ignore the registers  $L$  and  $M$  as they have already been measured or traced-out.
- 2 Let  $\ell$  be the width of the  $N$  register of the synthesis  $S$ . Define the stopping width  $w_0 \equiv 20d(\Delta + h(n) + 2)$ .
- 3 **if**  $\ell < w_0 = 20d(\Delta + h(n) + 2)$  **OR**  $\eta < 1$  **then**
- 4     Compute the quantity  $\langle 0_N | \phi_S | 0_N \rangle$  to within error  $\epsilon$ .
- 5     **return**  $\mathcal{A}_{full}(S, \mathcal{B}, \epsilon, D - 1)$
- 6 **else**
- 7     We will “slice” the  $D$ -Dimensional geometrically-local, depth- $d$  synthesis  $S$  in  $\Delta$  different locations, as follows:
- 8     Since  $N$  is  $D$ -Dimensional we define a region  $Z \subset N$  to be the sub-hyper-cube of  $N$  which has width  $10d(\Delta + h(n) + 2)$ , and is centered at the halfway point of  $N$  width-wise (about the point  $\ell/2$  of the way across  $N$ ). Since the maximum amount of width collectively between  $\Delta$  slices in  $K_{heavy}$  is  $10d\Delta + 10d \cdot h(n)$  (see Algorithm 1), we are guaranteed that the region  $Z$  will contain at least  $\Delta$  slices,  $K_1, K_2, \dots, K_\Delta$ , from  $K_{heavy}$ . For any two slices  $K_i, K_j \in K_{heavy}$ , let the un-normalized states  $|\phi_{L,i}\rangle, |\phi_{i,j}\rangle, |\phi_{j,R}\rangle$ , and corresponding sub-syntheses  $S_{L,i}, S_{i,j}, S_{j,R}$  be as defined in Definition 23 from [CC21], with  $K = \log^3(n)$ . We will use these to describe the result of our division step below.
- 9     For each  $K_i \in K_{heavy}$  pre-compute the quantity  $\kappa_{T,\epsilon}^i$ , with  $T = \log^3(n)$ , and  $\epsilon = \delta 2^{-10 \log(n) \log(\log(n))}$ .

10 **return**

$$\sum_{i=1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}(S_{i,R}, \eta - 1) \quad (2)$$

$$- \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1) \cdot \mathcal{A}(S_{j,R}, \eta - 1) \quad (3)$$

$$+ \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta - 1) \cdot \mathcal{A}(S_{j,R}, \eta - 1) \cdot \left[ \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \mathcal{A}_{full} \left( \left( \bigotimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \bigotimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K \right), \mathcal{B}, \frac{\epsilon}{2^\Delta}, D - 1 \right) \right] \quad (4)$$

11

---

Now that we have explicitly stated our main algorithm we can state and prove our main result,

which gives runtime bounds and error bounds for our main algorithm. Our algorithm is defined for any dimension  $D$ , and our main result is proved by induction on dimension  $D$ .

**Theorem 8.** *For any  $D$ -dimensional geometrically-local, constant depth quantum circuit  $C$  acting on  $n$  qubits, the algorithm  $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$  computes the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within  $\delta$  additive error in time  $\delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D \cdot 3^D})}$ . Furthermore, if we let  $w_{D+1}, w_{D+2}, \dots$  be the widths of the qubit array in dimensions  $D+1, D+2, \dots$  respectively. Then for any geometrically-local, constant depth quantum circuit  $C$  acting on a lattice of  $n$  qubits having side length at most  $w_{D+i}$  in dimension  $D+i$ , the algorithm  $\mathcal{A}_{full}(S = (C, L, M, N), \mathcal{B}, \delta, D)$  computes the quantity  $|\langle x | C | 0^{\otimes n} \rangle|^2$  to within  $\delta$  additive error in time  $\delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D \cdot 3^D} w^{1/3})}$ , where  $w \equiv \prod_{i=1}^{\infty} w_{D+i}$ .*

*Proof.* We will prove Theorem 8 by induction on the dimension  $D$ . For the base-case,  $D = 3$ , this theorem is a direct consequence of the main result of [CC21]. For  $D > 3$ , assuming, by induction, that we have already established Theorem 8 for dimension  $D - 1$ , the dimension  $D$  version of the Theorem follows by the runtime analysis and error analysis in subsections 3.1 and 3.2 respectively. The key inductive step in those two analyses happens at the point in the analysis where Algorithm 2 makes calls, such as  $\mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1)$ , to a  $D - 1$  dimensional version of Algorithm 1. At those points the runtime and error guarantees for the  $D - 1$  dimensional version of  $\mathcal{A}_{full}$  that are required by the analyses in subsections 3.1 and 3.2 are ensured by the inductive assumption that Theorem 8 already holds for the  $D - 1$  dimensional case.  $\square$

### 3.1 Run-time Analysis

Let  $T_1(l, D, d, w, \delta)$  represents the run-time of algorithm 1 for a problem with side length  $l$  in dimension  $D$  with circuit depth  $d$  and thickness  $w$  in dimensions  $> D$  to error  $\delta$ . Let  $T_2(l, D, d, w, \epsilon)$  be represent the same for algorithm 2. Then we may bound  $T_1$  as follows:

$$T_1(l, D, d, w, \delta) < \frac{n^{1/D}}{10d} T_1(l, D - 1, d, O(wd), E_1(\delta)) + T_2(l, D, d, w, E_2(\delta)),$$

$$T_1(l, 2, d, w, \delta) = B(n, d, w, \delta)$$

where  $E_1(\delta) = 2^{\frac{\log(\delta)}{2h(n)} - 1} - 2^{\frac{\log(\delta)}{h(n)} - 1}$  and  $E_2(\delta) = \delta 2^{-10 \log(n) \log(\log(n))}$ .

The term  $\frac{n^{1/D}}{10d} T_1(l, D - 1, d, O(wd), E_1(\delta))$  follows from lines 6-10 of algorithm 1. This entails making  $\frac{n^{1/D}}{10d}$  calls of algorithm 1 on a depth  $d$  synthesis in  $D - 1$  dimensions to error  $E_1(\delta)$  with thickness  $O(d)$  in dimension  $D$ . See the analysis of Theorem 28 of [CC21] for details on how this sub-problem is constructed. The term  $T_2(l, D, d, w, E_2(\delta))$  refers to the call of algorithm 2 made in line 14 of algorithm 1. The base case follows directly from line 5 of algorithm 1. By standard recursion analysis, we get that

$$T_1(l, D, d, w, \delta) < n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) + \sum_{i=0}^{D-3} n^{\frac{i}{D}} T_2(l, D - i, d, O(wd^i), E_2(E_1^{(i)}(\delta)))$$

where the notation  $E_1^{(i)}(\delta)$  refers to applying  $E_1$  to  $\delta$   $i$  times.

Similarly, we can bound  $T_2$  as follows:

$$T_2(l, D, d, w, \epsilon) < 2\Delta T_2(\frac{3}{4}l, D, d, w, \epsilon) + \Delta^2 T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon)$$

$$+ \Delta^2 2^\Delta T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon)) + 2\Delta T_1(l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon) + \text{poly}(n)$$



$$T_2(O(1), D, d, w, \epsilon) = T_1(l, D - 1, d, O(w), \epsilon)$$

where  $E_3(\epsilon) = \frac{\epsilon}{2^\Delta}$

The term  $2\Delta T_2(\frac{3}{4}l, D, d, w, \epsilon)$  follows from the calls to  $\mathcal{A}(S_{L,i}, \eta - 1)$  and  $\mathcal{A}(S_{i,R}, \eta - 1)$  for each  $i$ . The term  $\Delta^2 T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon)$  refers to the calls to  $\mathcal{A}_{full}(S_{i,j}, \mathcal{B}, \epsilon, D - 1)$  for each  $i$  and  $j > i$ . The term  $\Delta^2 2^\Delta T_1(l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon))$  refers to the calls to  $\mathcal{A}_{full}\left(\left(\otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \otimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \mathcal{B}, \frac{\epsilon}{2^\Delta}, D - 1\right)$  for each  $i, j > i$ , and  $\sigma$ . The term  $2\Delta T_1(l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon)$  refers to the calculation of  $\kappa_{T,\epsilon}$  for each  $i$ . For details regarding the construction of the sub-problems for the last three terms, refer to the run-time analysis of algorithm 2 of [CC21]. The final  $\text{poly}(n)$  term follows from the calculation of the region  $Z$  detailed in line 8 of algorithm 2. The base case follows from the fact that if we have a problem in  $D$  dimensions with an  $O(1)$  sized edge, we may apply an algorithm in  $D - 1$  dimensions to solve it at the cost of an extra  $O(1)$  sized thickness. By standard recursion analysis, we get that

$$\begin{aligned} T_2(l, D, d, w, \epsilon) &< (2\Delta)^\eta \cdot T_2\left(\left(\frac{3}{4}\right)^\eta l, D, d, w, \epsilon\right) + \sum_{i=0}^{\eta-1} (2\Delta)^i (\Delta^2 T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^3 \text{polylog}(n), O(wd), \epsilon\right) \\ &+ \Delta^2 2^\Delta T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^3 \text{polylog}(n), O(wd), E_3(\epsilon)\right) + 2\Delta T_1\left(\left(\frac{3}{4}\right)^i l, D - 1, d^2 \text{polylog}(n), O(wd), \epsilon\right) + \text{poly}(n)) \end{aligned}$$

Now, as we begin to substitute the recurrence relation for  $T_2$  (in terms of  $T_1$ ) into the recurrence relation for  $T_1$  (in terms of  $T_2$ ), we need to define  $\eta_i$ , the number of recursive calls made by  $T_2$  to  $T_1$  in the  $i$ -th dimension. Let us define  $\eta_i$  as the following:

$$\eta_i = \log_{3/4}(n^{-1/(i)}) = \frac{\log(n)}{(i) \log(4/3)} \quad (5)$$

Now that we have defined  $\eta_i$ , let us substitute the  $T_2$  recurrence relation into  $T_1$ :

$$\begin{aligned} T_1(l, D, d, w, \delta) &< n^{\frac{D-2}{D-1}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ &+ \sum_{i=0}^{D-3} n^{\frac{i}{D}} (2\Delta)^{\eta_{D-1}} \cdot \left[ T_1\left(l, D - i - 1, d, O(wd^i), E_2(E_1^{(i)}(\delta))\right) \right. \\ &+ \sum_{j=0}^{\eta_{D-i-1}} (2\Delta)^j \left( \Delta^2 T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^3 \text{polylog}(n), O(wd^{i+1}), E_2(E_1^{(i)}(\delta))\right) \right. \\ &+ \Delta^2 2^\Delta T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta)))\right) \\ &\left. \left. + 2\Delta T_1\left(\left(\frac{3}{4}\right)^j l, D - i - 1, d^2 \text{polylog}(n), O(wd^{i+1}), E_2(E_1^{(i)}(\delta))\right) + \text{poly}(n) \right) \right] \end{aligned}$$

where the first  $T_1$  term on the right-hand side comes from unrolling the  $T_2$  term in  $T_2$ 's recurrence relation down to its base-case.



We can then continue to simplify the upper bound by combining the three terms in the second summation term into  $3\Delta^2 2^\Delta T_1 \left( \left(\frac{3}{4}\right)^j l, D-i-1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta))) \right)$  since  $3\Delta^2 2^\Delta \geq (2\Delta + \Delta^2 2^\Delta + \Delta^2)$  and  $E_3(E_2(E_1^{(i)}(\delta))) \geq E_2(E_1^{(i)}(\delta))$ .

$$\begin{aligned} T_1(l, D, d, w, \delta) &< n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ &+ \sum_{i=0}^{D-3} n^{\frac{i}{D}} (2\Delta)^{\eta_{D-i}} \cdot \left[ T_1 \left( l, D-i-1, d, O(wd^i), E_2(E_1^{(i)}(\delta)) \right) \right. \\ &\left. + \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^j \left( 3\Delta^2 2^\Delta T_1 \left( \left(\frac{3}{4}\right)^j l, D-i-1, d^3 \text{polylog}(n), O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta))) \right) \right) \right] \end{aligned}$$

Next, we can unpack the bracket in the first summation term to get the following:

$$\begin{aligned} T_1(l, D, d, w, \delta) &< n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ &+ \sum_{i=0}^{D-3} n^{\frac{i}{D}} (2\Delta)^{\eta_{D-i}} T_1 \left( l, D-i-1, d, O(wd^i), E_2(E_1^{(i)}(\delta)) \right) \\ &+ \sum_{i=0}^{D-3} \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^{j+\eta_{D-i}} n^{\frac{i}{D}} \left( 3\Delta^2 2^\Delta T_1 \left( \left(\frac{3}{4}\right)^j l, D-i-1, d^3 \text{polylog}(n), \right. \right. \\ &\left. \left. O(wd^{i+1}), E_3(E_2(E_1^{(i)}(\delta))) \right) \right) \end{aligned}$$

The following expression can be obtained by extracting the  $T_1$  terms from the summation terms. We do that by bounding all the  $T_1$  terms in the first summation term by  $T_1 \left( l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right)$  since the runtime will be longer when we start on higher dimension  $D$  instead of dimension  $D-i-1$ , larger thickness  $O(wd^D)$  instead of thickness  $O(wd^i)$ , and smaller error  $E_2(E_1^{(D)}(\delta))$  instead of  $E_2(E_1^{(i)}(\delta))$ . A similar argument could be made for the  $T_1$  terms in the second summation term.

$$\begin{aligned} T_1(l, D, d, w, \delta) &< n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) \\ &+ T_1 \left( l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right) \sum_{i=0}^{D-3} n^{\frac{i}{D}} (2\Delta)^{\eta_{D-i}} \\ &+ 3\Delta^2 2^\Delta T_1 \left( l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) \sum_{i=0}^{D-3} \sum_{j=0}^{\eta_{D-i}-1} (2\Delta)^{j+\eta_{D-i}} n^{\frac{i}{D}} \end{aligned}$$

In the following step, for the first summation term, we bound the  $n^{\frac{i}{D}}$  term by  $\text{poly}(n)$  and the  $(2\Delta)^{\eta_{D-i}}$  by  $2^{\text{polylog}(n)}$  since  $\Delta, \eta = O(\log(n))$ . Since we have  $O(D)$  terms in the first summation term, we get  $O(D \text{poly}(n) 2^{\text{polylog}(n)})$ . Likewise, we can do the same thing for the second summation term to get the same upper bound.

$$T_1(l, D, d, w, \delta) < n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta))$$

$$\begin{aligned}
& + T_1 \left( l, D-1, d, O(wd^D), E_2(E_1^{(D)}(\delta)) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)}) \\
& + 3\Delta^2 2^\Delta T_1 \left( l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)})
\end{aligned}$$

The following expression can be obtained by combining the second and third term in the previous expression. We get  $T_1 \left( l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right)$  since  $d^3 \text{polylog}(n) > d$  and  $E_3(E_2(E_1^{(i)}(\delta))) \geq E_2(E_1^{(i)}(\delta))$  which would give us a larger runtime bound. The  $3\Delta^2 2^\Delta$  can be absorbed into the  $O(D \text{poly}(n) 2^{\text{polylog}(n)})$  term.

$$\begin{aligned}
T_1(l, D, d, w, \delta) & < n^{\frac{D-2}{D}} \mathcal{B}(l, d, O(wd^D), E_1^{(D-2)}(\delta)) \\
& + T_1 \left( l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)})
\end{aligned}$$

Now, we substitute the BGM algorithm's runtime from Theorem 5 of [?] into the first term to get the recurrence for  $T_1$  for a certain dimension  $D$  in terms of the  $T_1$  for one dimension lower.

$$\begin{aligned}
T_1(l, D, d, w, \delta) & < \text{poly}(n) (E_1^{(D-2)}(\delta))^{-2} 2^{d^3 w^{1/D}} \\
& + T_1 \left( l, D-1, d^3 \text{polylog}(n), O(wd^D), E_3(E_2(E_1^{(D)}(\delta))) \right) O(D \text{poly}(n) 2^{\text{polylog}(n)})
\end{aligned}$$

Before we begin to unroll the recurrence relation for  $T_1$  with respect to its dimension, let us first define  $f(d)$ , the depth of the block-encoding, and  $f^{(k)}(d)$ , the depth of the block-encoding after unrolling the recurrence relation for  $k$  dimensions

$$\begin{aligned}
f(d) & = d^3 \text{polylog}(n) \\
f^{(k)}(d) & < d^{3^k} (\text{polylog}(n))^{3^k}
\end{aligned}$$

We can also define  $g(d, w)$ , the thickness of the circuit, and  $g^{(k)}(d)$ , the thickness of the circuit after unrolling the recurrence relation for  $k$  dimensions as follows:

$$\begin{aligned}
g(d, w) & = O(wd^D) \\
g^{(k)}(d, w) & = g^{(k-1)}(d, w) (f^{(k-1)}(d))^D \\
& = g^{(k-2)}(d, w) (f^{(k-2)}(d))^D (f^{(k-1)}(d))^D \\
& = g^{(k-\ell)}(d, w) \prod_{i=1}^{\ell} (f^{(k-i)}(d))^D \\
& = g(d, w) \prod_{i=1}^{k-1} (f^{(k-i)}(d))^D \\
& < O(wd^D) \left( \prod_{i=1}^{k-1} (d \text{polylog}(n))^{3^i} \right)^D \\
& < O(wd^D) (d \text{polylog}(n))^{D 3^k}
\end{aligned}$$

Now, we want to write the unrolling of the recurrence relation for  $T_1$  with respect to dimensions in terms of  $f(d)$  and  $g(d, w)$ . To simplify the writing, we define  $E_5(\delta) = E_3(E_2(E_1^{(D)}(\delta)))$ .

The following expression is obtained by unrolling  $T_1$ 's recurrence relation for an arbitrary dimension  $D$  to dimension 2 which is the base-case for  $T_1$ .

$$T_1(\ell, D, d, w, \delta) < O((D \text{poly}(n) 2^{\text{polylog}(n)})^{D-2}) T_1(\ell, 2, f^{(D-2)}(d), g^{(D-2)}(d, w), E_5^{(D-2)}(\delta)) \\ + \sum_{i=0}^{D-3} O((D \text{poly}(n) 2^{\text{polylog}(n)})^i) \text{poly}(n) \left( E_1^{(D-i-2)} \left( E_5^{(i)}(\delta) \right) \right)^{-2} 2^{(f^{(i)}(d))^3 (g^{(i)}(d, w))^{\frac{1}{D-i}}}$$

To simplify the upper bound for  $T_1(l, D, d, w, \delta)$ , we wrote the terms in the sum in their maximum form where  $i = D$  for the big- $O$  term in the sum as well as the number of compositions for  $E_1$ ,  $E_5$ ,  $f(d)$ , and  $g(d, w)$ . Note that the more we compose  $E_1$  and  $E_5$ , the smaller they get and hence their inverse-squared form will be larger. For  $f(d)$  and  $g(d, w)$ , the more we composed them, the greater the depth and the thicker the thickness of the block-encoding which gives us an upper bound for the runtime. Note how we chose the upper bound for the exponent of the  $g^{(D)}(d, w)$  to be  $\frac{1}{3}$  to get the smallest root form to maximize the exponent of the 2 term.

$$T_1(\ell, D, d, w, \delta) < O((D \text{poly}(n) 2^{\text{polylog}(n)})^D) T_1(\ell, 2, f^{(D)}(d), g^{(D)}(d, w), E_5^{(D)}(\delta)) \\ + D \cdot O((D \text{poly}(n) 2^{\text{polylog}(n)})^D) \text{poly}(n) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}}$$

Next, we write the first term of the right-hand side of the first inequality according to BGM's runtime as given in Theorem 5 of [?] and then brought the  $D \cdot \text{poly}(n)$  coefficient in the second term into the second term's big- $O$  term.

$$T_1(\ell, D, d, w, \delta) < O((\text{poly}(n))^{D+1} (D 2^{\text{polylog}(n)})^D) (E_5^{(D)}(\delta))^{-2} 2^{(f^{(D)}(d))^2 (g^{(D)}(d, w))^{1/D}} \\ + O((D \text{poly}(n))^{D+1} (2^{\text{polylog}(n)})^D) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}} \\ < O((D \text{poly}(n))^{D+1} (2^{\text{polylog}(n)})^D) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} 2^{(f^{(D)}(d))^3 (g^{(D)}(d, w))^{\frac{1}{3}}}$$

where the second inequality comes from the fact that first term is smaller than the second term and hence can be absorbed into the second term.

Now, we substitute the upper bounds for  $f^{(k)}(d)$  and  $g^{(k)}(d, w)$  as previously defined into the above expression to get the following inequality:

$$T_1(\ell, D, d, w, \delta) < O((D \text{poly}(n))^{D+1} (2^{\text{polylog}(n)})^D) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{(d \text{polylog}(n))^{3D+1} (O(w d^D)^{\frac{1}{3}} (d \text{polylog}(n))^{D3D-1})} \\ = O((D \text{poly}(n))^{D+1} (2^{\text{polylog}(n)})^D) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{O(w d^D)^{\frac{1}{3}} (d \text{polylog}(n))^{(9+D)3D-1}} \\ < O((D \text{poly}(n))^{D+1} (2^{\text{polylog}(n)})^D) \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} \\ \cdot 2^{O(d^{(3D+1+D/3+D3D-1)}) (\text{polylog}(n))^{(3D+1+D3D-1)} w^{1/3}}$$

$$\begin{aligned}
&< \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} \cdot 2^{O(d^{(3^{D+1}+D/3+D3^{D-1})}(\text{polylog}(n))^{(3^{D+1}+D3^{D-1})}w^{1/3})} \\
&< \left( E_1^{(D)} \left( E_5^{(D)}(\delta) \right) \right)^{-2} \cdot 2^{O((d\text{polylog}(n))^{D3^D}w^{1/3})}
\end{aligned}$$

Now note that

$$E_1(\delta) = 2^{\frac{\log(\delta)}{2h(n)}-1} - 2^{\frac{\log(\delta)}{h(n)}-1} = \frac{1}{2} \left( 2^{\frac{\log(\delta)}{2h(n)}} - 2^{\frac{\log(\delta)}{h(n)}} \right) \geq \frac{\ln 2}{2} 2^{\frac{\log(\delta)}{h(n)}} \left( \frac{\log(\delta)}{2h(n)} - \frac{\log(\delta)}{h(n)} \right) = -\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2$$

Hence, by monotonicity,

$$\begin{aligned}
E_1(E_1(\delta)) &= -\frac{\log(E_1(\delta))}{4h(n)} 2^{\frac{\log(E_1(\delta))}{h(n)}} \cdot \ln 2 \geq -\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{h(n)}} \cdot \ln 2 \\
&= -\frac{\log\left(-\frac{\log(\delta)}{4h(n)}\right) + \log\left(2^{\frac{\log(\delta)}{h(n)}}\right) + \log(\ln 2)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{h(n)}} \cdot \ln 2 \\
&\geq -\frac{\log(\ln 2)}{4h(n)} 2^{\frac{\log\left(-\frac{\log(\delta)}{4h(n)} 2^{\frac{\log(\delta)}{h(n)}} \cdot \ln 2\right)}{h(n)}} \cdot \ln 2 \\
&= -\frac{\log(\ln 2)}{4h(n)} 2^{\frac{\log(E_1(\delta))}{h(n)}} \cdot \ln 2 \\
&\geq -\frac{\log(\ln 2)}{4h(n)} (E_1(\delta))^{\frac{1}{h(n)}} \cdot \ln 2 \\
&\geq -\frac{\log(\ln 2)}{4h(n)} E_1(\delta) \cdot \ln 2
\end{aligned}$$

And so for some constant  $a$  we get,

$$E_1(a\delta) \geq -\frac{\log(\ln 2)}{4h(n)} a\delta \cdot \ln 2$$

Therefore

$$\begin{aligned}
E_1^{(D)}(\delta) &\geq \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{D-1} E_1(\delta) \geq \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^D \delta \\
\implies E_5(\delta) &= E_3(E_2(E_1^{(D)}(\delta))) \geq 2^{-10\log(n)\log(\log(n))-\Delta} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^D \delta \\
\implies E_5^{(D)}(\delta) &\geq 2^{D(-10\log(n)\log(\log(n))-\Delta)} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{D^2} \delta \\
\implies (E_1^{(D)} \circ E_5^{(D)})(\delta) &\geq 2^{D(-10\log(n)\log(\log(n))-\Delta)} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{D^2+D} \delta
\end{aligned}$$

Thus with  $\Delta = \log(n)$  we get that

$$\left( (E_1^{(D)} \circ E_5^{(D)})(\delta) \right)^{-2} \leq 2^{D(10\log(n)\log(\log(n))+\Delta)} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{-2D^2-2D} \delta^{-2}$$

$$= 2^{D \text{polylog}(n)} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{-2D^2-2D} \delta^{-2}$$

Plugging this into our run-time bound we get

$$\begin{aligned} T_1(\ell, D, d, w, \delta) &< 2^{D \text{polylog}(n)} \left( \frac{-\log(\ln(2)) \ln(2)}{4h(n)} \right)^{-2D^2-2D} \delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D^3} w^{1/3})} \\ &= \delta^{-2} \cdot 2^{O((d \text{polylog}(n))^{D^3} w^{1/3})} \end{aligned}$$

## 3.2 Error Analysis

### 3.2.1 Error Analysis for Algorithm 1

The error analysis of the error obtained by  $\mathcal{A}_{full}(S, \mathcal{B}, \delta, D)$  can be broken into four cases according to the IF statements on Lines 1, 3, 11, and 12 of Algorithm 1. The first three cases can be easily shown to return the value in  $\delta$ -additive error within the promised runtime as shown in page 16 and 20 of [CC21].

In the event that Line 12 is satisfied, Algorithm 1 returns the following quantity:

$$\mathcal{A}(S, \eta = \frac{\log(n)}{D \log(4/3)}, \Delta = \log(n), \epsilon = \delta 2^{-10 \log(n) \log(\log(n))}, h(n) = \log^7(n), K_{heavy}, D, \mathcal{B}),$$

which we know is an  $f(S, \eta_D, \Delta, \epsilon, D)$ -additive error approximation of  $|\langle 0_{ALL} | C | 0_{ALL} \rangle|^2$ . Recall the definition of  $\eta_D$  defined in Equation 5 and with  $\eta_D = \frac{\log(n)}{D \log(4/3)}$ , by Lemma 32 from [CC21] we know that:

$$\begin{aligned} f(S, \eta, \Delta, \epsilon) &\leq \eta_D (20\Delta^2)^{\eta_D} \left( (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) \\ &= \eta_D (20\Delta^2)^{\eta_D} 3\Delta^2 O(\mathcal{E}_3(n, K, T, \epsilon, \Delta)) \\ &= \eta_D (20\Delta^2)^{\eta_D} 3\Delta^2 O\left(2^\Delta (2e(n))^K + 2^\Delta K(e(n)^{2T} + \epsilon) + \epsilon\right) \\ &= \frac{\log(n)}{D \log(4/3)} \left(20 \log^2(n)\right)^{\frac{\log(n)}{D \log(4/3)}} 3 \log^2(n) O\left(2^{\log(n)} (2(1 - 2^{\frac{\log(\delta)}{\log^7(n)}})) \log^3(n) \right. \\ &\quad \left. + 2^{\log(n)} \log^3(n) \left( (1 - 2^{\frac{\log(\delta)}{\log^7(n)}})^{2 \log^3(n)} + \epsilon \right) + \delta 2^{-10 \log(n) \log(\log(n))} \right) \\ &\leq (\log(n))^{2 \log(n)} \cdot \text{poly}(n) \cdot \left( (2(1 - 2^{\frac{\log(\delta)}{\log^7(n)}}))^{\log^3(n)} + \epsilon + \delta 2^{-10 \log(n) \log(\log(n))} \right) \\ &\leq (\log(n))^{2 \log(n)} \cdot \text{poly}(n) \cdot \left( \left( O\left(\frac{1}{\log^4(n)}\right) \right)^{\log^3(n)} + 2 \cdot \delta 2^{-10 \log(n) \log(\log(n))} \right) \\ &\leq 2^{2 \log(n) \log(\log(n))} \cdot \text{poly}(n) \cdot \left( O\left(\frac{1}{\log^4(n)}\right) \right)^{\log^3(n)} + \delta 2^{-8 \log(n) \log(\log(n))} \\ &\leq o(1) \cdot \delta + o(1) \cdot \delta = o(1) \cdot \delta \end{aligned} \tag{6}$$

where the first inequality follows from our result from the next subsection and the rest follows by calculation, noting that  $E_3(n, K, T, \epsilon, \Delta) \geq (2e(n) + 2g(n))^\Delta$  for our specific choice of parameters (in particular  $\Delta = \log(n)$ ). Note from [CC21] that  $e(n) \leq (1 - 2^{\frac{\log(\delta)}{\log^7(n)}}) = O(1/\log^4(n))$  (since  $\delta \geq n^{-\log^2(n)} = 2^{-\log(n)^3}$  as verified in Algorithm 1),  $K = \log^3(n)$ ,  $T = \log^3(n)$ , and  $\epsilon = \delta 2^{-10 \log(n) \log(\log(n))}$ . The final inequality, which claims  $2^{2 \log(n) \log(\log(n))} \cdot \text{poly}(n) \cdot \left(O\left(\frac{1}{\log^4(n)}\right)\right)^{\log^4(n)} = o(1) \cdot \delta$ , again follows because  $\delta \geq n^{-\log^2(n)}$  as verified in the driver algorithm, Algorithm 1.

### 3.2.2 Error Analysis for Algorithm 2

As described on page 22 of [CC21],  $\langle 0_{ALL} | \Psi_\emptyset \rangle \langle \Psi_\emptyset | 0_{ALL} \rangle$  is the quantity that we wish for Algorithm 2 to output. Refer to Definition 17 and Lemma 18 from [CC21] for the definition of  $|\Psi_\emptyset\rangle$  and  $|\Psi_\sigma\rangle$  for the subsequent analysis. Since Algorithm 2 depends on recursively calling itself, recall  $\eta_i$  from Equation 5 that defines the number of recursive calls for some dimension  $i$ . The error between the returned output of Algorithm 2, (defined on Line 10 of that algorithm) and the desired output quantity  $\langle 0_{ALL} | \Psi_\emptyset \rangle \langle \Psi_\emptyset | 0_{ALL} \rangle$  is written below:

$$f(S, \eta_D, \Delta, \epsilon, D) \leq \left\| \langle 0_{ALL} | \Psi_\emptyset \rangle \langle \Psi_\emptyset | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\| \quad (7)$$

$$\leq \left\| \langle 0_{ALL} | \Psi_\emptyset \rangle \langle \Psi_\emptyset | 0_{ALL} \rangle - \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle \right\| \quad (8)$$

$$+ \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\| \quad (9)$$

$$\leq (2e(n) + 2g(n))^\Delta + \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle - \mathcal{A}(S, \eta_D, D, \epsilon) \right\|$$

$$= (2e(n) + 2g(n))^\Delta + \left\| \sum_{\sigma \in \mathcal{P}([\Delta]) \setminus \emptyset} (-1)^{|\sigma|+1} \langle 0_{ALL} | \Psi_\sigma \rangle \langle \Psi_\sigma | 0_{ALL} \rangle \right.$$

$$- \left( \sum_{i=1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) \right.$$

$$- \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \eta_{D-1}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon)$$

$$+ \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon)$$

$$\cdot \left[ \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \mathcal{A}_{full} \left( \left( \bigotimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \bigotimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \eta_{D-1}, D - 1, E_3(\epsilon) \right) \right] \right\| \quad (10)$$

Grouping analogous terms and using triangle inequality gives:

$$f(S, \eta_D, \Delta, \epsilon, D) \leq (2e(n) + 2g(n))^\Delta + \left\| \sum_{i=1}^{\Delta} \left( \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) - \langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle \right) \right\|$$

$$\begin{aligned}
& - \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \eta_{D-1}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \\
& - \left. \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right) + \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \right. \\
& \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \cdot (-1)^{|\sigma|+1} \mathcal{A}_{full} \left( \left( \otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \otimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \eta_{D-1}, D - 1, E_3(\epsilon) \right) \\
& - \left. \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right) \Big\| \\
& \leq (2e(n) + 2g(n))^{\Delta} \\
& + \sum_{i=1}^{\Delta} \left\| \left( \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) - \langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle \right) \right\| \\
& + \sum_{i=1}^{\Delta} \sum_{j=i+1}^{\Delta} \left\| \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \eta_{D-1}, D - 1, E_3(\epsilon)) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& - \left. \left. \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right) \right\| \\
& + \sum_{i=1}^{\Delta} \sum_{j=i+2}^{\Delta} \left\| \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& \cdot \mathcal{A}_{full} \left( \left( \otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \otimes_{k \in \sigma} | 0_{M_k} \rangle \Pi_{F_k}^K \right), \eta_{D-1}, D - 1, E_3(\epsilon) \right) - \left. \left. \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right) \right\| \\
& \tag{11}
\end{aligned}$$

Now, we will adapt Lemma 29, 30, and 31 from [CC21] to generate our own version of those lemmas to bound the last three terms of the above inequality. Because their bounds are independent of dimensions, the proofs for the three lemmas below will be similar to the proofs in [CC21].

**Lemma 9.**

$$\begin{aligned}
& \left\| \left( \frac{1}{(\kappa_{T,\epsilon}^i)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{i,R}, \eta_D - 1, D, \epsilon) - \langle 0_{ALL} | \Psi_{\{i\}} \rangle \langle \Psi_{\{i\}} | 0_{ALL} \rangle \right) \right\| \\
& \leq \mathcal{E}_1(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D),
\end{aligned}$$

where  $\mathcal{E}_1(n, K, T, \epsilon) \equiv 10K(e(n)^{2T} + 6g(n) + \epsilon)$ .

**Lemma 10.**

$$\begin{aligned}
& \left\| \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}_{full}(S_{i,j}, \eta_{D-1}, D - 1, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& - \left. \left. \langle 0_{ALL} | \Psi_{\{i,j\}} \rangle \langle \Psi_{\{i,j\}} | 0_{ALL} \rangle \right) \right\| \\
& \leq \mathcal{E}_2(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D),
\end{aligned} \tag{12}$$

where  $\mathcal{E}_2(n, K, T, \epsilon) \equiv 10K(e(n)^{2T} + 6g(n) + \epsilon) + \epsilon$



**Lemma 11.**

$$\begin{aligned}
& \left\| \sum_{\sigma \in \mathcal{P}(\{i+1, \dots, j-1\}) \setminus \emptyset} (-1)^{|\sigma|+1} \left( \frac{1}{(\kappa_{T,\epsilon}^i \kappa_{T,\epsilon}^j)^{4K+1}} \mathcal{A}(S_{L,i}, \eta_D - 1, D, \epsilon) \cdot \mathcal{A}(S_{j,R}, \eta_D - 1, D, \epsilon) \right. \right. \\
& \cdot \mathcal{A}_{full} \left( \left( \otimes_{k \in \sigma} \Pi_{F_k}^K \langle 0_{M_k} | \right) \phi_{i,j} \left( \otimes_{k \in \sigma} |0_{M_k}\rangle \Pi_{F_k}^K \right), \eta_{D-1}, D-1, E_3(\epsilon) \right) \\
& \left. - \langle 0_{ALL} | \Psi_{\{i,j\} \cup \sigma} \rangle \langle \Psi_{\{i,j\} \cup \sigma} | 0_{ALL} \rangle \right\| \\
& \leq \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 16f(S, \eta_D - 1, \Delta, \epsilon, D),
\end{aligned} \tag{13}$$

where

$$\mathcal{E}_3(n, K, T, \epsilon, \Delta) \equiv O \left( 2^\Delta (6g(n)) + 2^\Delta K \left( e(n)^{2T} + \epsilon \right) + \epsilon \right)$$

Returning to where we left off in Equation 11, using all three of the above Lemmas, we have

$$\begin{aligned}
f(S, \eta_D, \Delta, \epsilon, D) & \leq (2e(n) + 2g(n))^\Delta + \Delta (\mathcal{E}_1(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \quad + \Delta^2 (\mathcal{E}_2(n, K, T, \epsilon) + 2f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \quad + \Delta^2 (\mathcal{E}_3(n, K, T, \epsilon, \Delta) + 16f(S, \eta_D - 1, \Delta, \epsilon, D)) \\
& \leq (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 20\Delta^2 f(S, \eta_D - 1, \Delta, \epsilon, D) \\
& = (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \\
& \quad + 20\Delta^2 \left[ (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) + 20\Delta^2 f(S, \eta_D - 2, \Delta, \epsilon, D) \right] \\
& = \sum_{i=0}^{\eta_D-1} \left[ (20\Delta^2)^i \left( (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) \right] + (20\Delta^2)^{\eta_D} f(S, 0, \Delta, \epsilon, D) \\
& \leq \eta_D (20\Delta^2)^{\eta_D} \left( (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) + (20\Delta^2)^{\eta_D} \epsilon \\
& \leq \eta_D (20\Delta^2)^{\eta_D} \left( \epsilon + (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right) \\
& \leq \eta_D (20\Delta^2)^{\eta_D} \left( (2e(n) + 2g(n))^\Delta + 3\Delta^2 \mathcal{E}_3(n, K, T, \epsilon, \Delta) \right)
\end{aligned} \tag{14}$$

where the above inequalities follow because  $\mathcal{E}_3(n, K, T, \epsilon, \Delta) \geq \mathcal{E}_2(n, K, T, \epsilon) \geq \mathcal{E}_1(n, K, T, \epsilon)$  and  $f(S, 0, \Delta, \epsilon, \cdot) \leq \epsilon \leq \mathcal{E}_3(n, K, T, \epsilon, \Delta)$

## Acknowledgment

MC thanks Sergey Bravyi for helpful discussions. We thank Gorjan Alagic and Nolan Coble for attending and contributing to some project group meetings.

## References

- [BGM20] Sergy Bravyi, David Gosset, and Ramis Movassagh. Classical algorithms for quantum mean values. QIP, 2020. URL: <https://arxiv.org/abs/1909.11485>.
- [CC21] Nolan J. Coble and Matthew Coudron. Quasi-polynomial time approximation of output probabilities of geometrically-local, shallow quantum circuits. In *62nd Annual Symposium on Foundations of Computer Science, FOCS 2021*, 2021.
- [GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. STOC, 2019. URL: <https://arxiv.org/pdf/1806.01838.pdf>.
- [KMM21] Yasuhiro Kondo, Ryuhei Mori, and Ramis Movassagh. Fine-grained analysis and improved robustness of quantum supremacy for random circuit sampling, 2021. [arXiv: 2102.01960](https://arxiv.org/abs/2102.01960).
- [Mov20] Ramis Movassagh. Quantum supremacy and random circuits. QIP, 2020. URL: <https://arxiv.org/pdf/1909.06210.pdf>.
- [TD04] Barbara M. Terhal and David P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. *Quantum Inf. Comput.*, 4(2):134–145, 2004. doi:10.26421/QIC4.2-5.