Topological obstructions to implementing controlled unknown unitaries

Zuzana Gavorová¹, Matan Seidel², and Yonathan Touati¹

¹School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel
²School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel

Abstract

Is a quantum algorithm capable of implementing an if-clause? Given a black-box subroutine, a d-dimensional unitary \( U \in U(d) \), a quantum if-clause would correspond to applying it to an input qudit if and only if the value of a control qubit is 1. It was previously only known that implementations using single access to the oracle \( U \) are impossible [TGMV13, AFCB14]. Our main result is a strong generalization of this impossibility result: we prove that there is no unitary oracle algorithm implementing control \( \phi(U) = |0\rangle\langle 0| \otimes 1 + e^{i\phi(U)} |1\rangle\langle 1| \otimes U \), for some \( U \)-dependent phase \( \phi(U) \), even if allowed any finite number of calls to \( U \) and \( U^\dagger \), and even if required to only approximate the desired operator control \( \phi(U) \). Even further, there is no such postselection oracle algorithm, i.e. a unitary oracle algorithm followed by a binary success/fail measurement, that reports success and implements control \( \phi(U) \) with a nonzero probability for each \( U \in U(d) \).

Our proof relies on topological arguments which can be viewed as a modification of the Borsuk-Ulam theorem. Combining the topological arguments with the algorithm of [DNSM19] in fact leads to an interesting dichotomy: implementing control \( \phi(U_m) \) in our model is possible if and only if the integer \( m \) is a multiple of the oracle’s dimension \( d \).

Our impossibility no longer holds if the model is relaxed, either by dropping the worst-case requirement that the algorithm works for all \( U \in U(d) \), or, remarkably, by allowing measurements more general than a binary postselection measurement. We observe that for both relaxations, inefficient algorithms exist, and it remains open whether efficient ones do.

1 Introduction

Many important quantum algorithms make use of the quantum if clause, a primitive which given a unitary \( U \in U(d) \) builds control \( \phi(U) = |0\rangle\langle 0| \otimes 1 + |1\rangle\langle 1| \otimes U \). For example, the famous algorithms for factoring integers [Sho94], solving linear systems of equations [HHL09], solving semidefinite programs [BS17, VAGGdW20], or characterising quantum many-body systems [TOV+11, WBAG11] all use quantum phase estimation [Kit95, CEMM98], which, in turn, requires calls to control \( \phi(U) \). If a classical description of \( U \) is available, as is the case in the fast algorithms mentioned above, there is the following general, implementation-independent way to build control \( \phi(U) \) from \( U \): Given its classical description, find \( U \)'s decomposition to elementary gates and add a control to each of them, then run the controlled elementary gates to implement control \( \phi(U) \) [BBC+95].

It is a very natural question whether a quantum circuit can implement control \( \phi(U) \) when treating \( U \) as an oracle [AK07]. One solution is immediately available; call \( U \) sufficiently many
times to apply process tomography [CN97, PCZ97], providing a classical description of $e^{i\alpha}U$ for some $\alpha \in [0, 2\pi]$ (the procedure must be insensitive to the global phase of $U$), then continue as described above - implementing control($e^{i\alpha}U$). Much simpler solutions exist in specific physical implementations; in optics [ZRK+11], computation with trapped ions [KDDB14] or superconducting qubits [FMKB15]. For example in optics, since the oracle gate $U$ occupies a certain physical space, an interferometer with polarising beamsplitters can control whether a photon passes through the gate $U$ or not depending on the polarisation degree of freedom. This implementation-dependent solution does not require measurements. We ask whether a unitary solution exists in a quantum circuit. Our answer is in the negative: A unitary quantum circuit implementing control($e^{i\alpha}U$) is impossible. Equivalently, in any quantum circuit with measurements that does implement control($e^{i\alpha}U$) (as the one corresponding to the process tomography approach), the measurements are necessary for disentangling the gained phase $\alpha$ from ancilla registers.

First we introduce the notion of task to simplify the discussion of preceding works and the formulation of our result. Several works studied the possibility of using oracle access to $U$ for implementing functions of $U$: the control function [TGMV13, AFCB14, DNSM19], but also complex conjugation [MSM19], transpose and inverse [QDS+19a, QDS+19b], raising $U$ to some fractional power [SMM09, GSLW19], and more. In all these works, one is interested in viewing the access to the subroutines $U$ as an oracle access, and treating the process achieving the desired function of $U$, $t(U)$, as an algorithm. Treating subroutines $U$ as oracles has a great deal of flexibility; replacing the subroutines by different ones (different $U$’s) preserves the functionality of the algorithm (namely, the function $t$). Since the algorithm itself may be used as a subroutine, the $t(U)$ it should achieve is an operator - i.e. the algorithm should implement $t(U)$. At the same time, it makes sense to specify what ways to access the oracle $U$ are available to the algorithm. To this end we define:

Definition 1 (Task). A task is a pair $(t, \Sigma)$, where

1. The task function $t : U(d) \rightarrow L(H_i)$ indicates that given an oracle $U \in U(d)$, we wish to implement the operator $t(U)$ to the task Hilbert space $H_i$.
2. The query alphabet $\Sigma$ is a set of functions on $U(d)$, such that if the oracle is $U \in U(d)$ and $\sigma \in \Sigma$, then the algorithm is allowed to query $\sigma(U)$. The set $\Sigma$ usually contains the identity $id : U \rightarrow U$.

where $L(H)$ is the set of linear operators from the finite-dimensional Hilbert space $H$ to itself.

Roughly, we say that an algorithm achieves the task $(t, \Sigma)$, if it implements $t(U)$ while accessing $U$ only via the functions in the query alphabet $\Sigma$. See Definitions 4 and 8 for more rigour. We distinguish worst-case algorithms, which implement $t(U)$ for all $U \in U(d)$, and average-case algorithms, which can fail for some $U \in U(d)$. All the algorithms we mention throughout this paper are worst-case unless we explicitly indicate otherwise.

We restate some of the previous works in this ‘task’ $(t, \Sigma)$ terminology. First of all, phase estimation is concerned with a task whose query alphabet contains control. Miyazaki et al. [MSM19] presented an algorithm for the task of complex conjugation $(t, \Sigma) = (U \mapsto U^*, \{id\})$. This was followed by algorithms by Quintino et al. [QDS+19b, QDS+19a] for transpose $(U \mapsto U^T, \{id\})$ and inversion $(inv : U \mapsto U^{-1}, \{id\})$. Sheridan et. al. [SMM09] presented the $q$-th power algorithm, an average-case algorithm achieving $(U \mapsto U^q, \{id, inv\})$ for any fixed $q \in \mathbb{R}$.

The specific question of implementing control($U$) for all $U \in U(d)$, and some variations of this question, have already been studied quite extensively before. Thompson et al. [TGMV13] observed that the (control, $\{id\}$) task is impossible; any algorithm that can implement control($U$) from calls to $U$ is unphysical - if applied to the correct input and followed by the correct measurement, it would give a physical process distinguishing $U$ from $-U$, contradicting the fact that a difference in global phase is physically indistinguishable. Araújo et al. [AFCB14] asked about implementing control up to a global phase on $U$, i.e. about the task (control$_\phi$, $\{id\}$)
with $\text{control}_\phi(U) = |0\rangle \langle 0| \otimes 1 + e^{i\phi(U)} |1\rangle \langle 1| \otimes U$ for any real function $\phi$. They proved that with one call to $U \in U(2)$ this task is impossible for a quantum circuit. Dong et al. [DNSM19] found an algorithm for $(U \mapsto \text{control}_\phi(U^d), \{\text{id}\})$ for $d$ the dimension of $U$. Together with the $\frac{1}{d}$-th power algorithm of Sheridan et al. [SMM09] these two results compose to an algorithm for $(\text{control}_\phi, \{\text{id}, \text{inv}\})$. Unfortunately, due to its [SMM09] component, this algorithm fails for some $U \in U(d)$.

1.1 Overview of Main Result

The main result of this paper is concerned with the worst-case achievability of the task $c-U := (\text{control}_\phi, \{\text{id}, \text{inv}\})$. We generalise the impossibility of Araújo et al. [AFCB14] to oracles of any fixed dimension $d$, $U \in U(d)$ and to any finite number of $\text{id}$ and $\text{inv}$ queries, and show that for any such unitary circuit $c-U$ is impossible. It remains impossible also if we add postselection, calling the resulting model postselection oracle algorithm. Most importantly, the impossibility holds even when one allows only approximate implementation of $\text{control}_\phi(U)$. This impossibility result should be contrasted with the above mentioned combination of the algorithms of Sheridan et al. [SMM09] and Dong et al. [DNSM19] which achieves the $c-U$ task for most $U$'s. We will also contrast it with the process tomography strategy. To get our impossibility, we prove a stronger result, regarding a more general task: $c-U^m := (U \mapsto \text{control}_\phi(U^m), \{\text{id}, \text{inv}\})$ for $m \in \mathbb{Z}$. We phrase our result as a dichotomy theorem regarding the possibility of this task, as a function of the relation between $m$ and $d$:

**Theorem 1 (The Exact Dichotomy).** Let $m \in \mathbb{Z}$ and let $d \in \mathbb{N}$ be the dimension of the oracle, $U \in U(d)$.
- If $d|m$ there exists a postselection oracle algorithm exactly achieving the task $c-U^m := (U \mapsto |0\rangle \langle 0| \otimes 1 + e^{i\phi(U)} |1\rangle \langle 1| \otimes U^m, \{\text{id}, \text{inv}\})$, for some $\phi : U(d) \to \mathbb{R}$.
- If $d \nmid m$ no such algorithm exists.

Since the $m = 1$ case of the task $c-U^m$, is the task $c-U$, the following impossibility, which is our main result, follows from our Dichotomy theorem:

**Corollary 1 (Main: Impossibility of Controlled $U$).** The task $c-U$ is impossible in the postselection oracle algorithm model.

**Theorem 2 (Roughly) (Impossibility of Approximate Controlled $U$).**

Theorem 1 and Corollary 1 can be generalized for the approximate setting.

The $d|m$ part of Theorem 1 is a corollary of Dong et al.’s [DNSM19] construction of the algorithm that achieves $c-U^d$. Our main contribution is the $d \nmid m$ direction, for which we prove the following topological lemma:

**Lemma 1.** Let $d \in \mathbb{N}$ and $m \in \mathbb{Z}$ such that there exists a function $f : U(d) \to S^1$ which is continuous and $m$-homogeneous, i.e. $f(\lambda U) = \lambda^m f(U)$ for each $\lambda \in S^1$. Then $m$ is a multiple of $d$.

Note that Lemma 1 is closely related to the Borsuk-Ulam theorem [Bor33]. For comparison, in Appendix A we use the Borsuk-Ulam theorem instead of Lemma 1 to prove Corollary 1 for even-dimensional oracles.
1.2 Corollaries of the dichotomy

Our results have interesting consequences for other tasks. For some corollaries, the exact dichotomy is enough. We have mentioned \[\text{QDS}^+19b, \text{QDS}^+19a\] who presented families of exact postselection oracle algorithms for the transpose and the inversion tasks. Our exact dichotomy implies that some of those algorithms implement \(U\)-dependent operations to their ancilla and are therefore less useful as subroutines. Another corollary deals with exact neutralisation, a task mentioned in [DNSM19]. The approximate dichotomy, on the other hand, can provide a new proof, in addition to the one of [SMM09], that a worst-case alternative to their fractional power algorithm is impossible.

1.3 Structure of the paper

The paper is structured as follows. Sections 2.1 to 2.4 define our model of computation and describe some of its properties. The remainder of Section 2 is needed only for the corollaries of the main result: Section 2.5 reviews Kitaev’s [Kit95] controlled-swap trick and its use in the \(q\)-th power algorithm [SMM09]. Section 2.6 reviews Dong et al.’s [DNSM19] algorithm for \(c\cdot U^d\), defining neutralisation along the way. In Section 3 we prove the main result - the exact dichotomy (Theorem 1). Then we discuss its consequences on neutralisation, inversion and transpose tasks. In Section 4 we generalise our dichotomy to the approximate case (Theorem 2). Section 5 discusses a corollary for the \(q\)-th power task, contrasting our worst-case impossibility with the average-case possibility. We also comment on the open question of an efficient average-case \(c\cdot U\) algorithm. Section 6 explains the remarkable fact that our model, while being quite natural, excludes the (worst-case) process-tomography strategy for \(c\cdot U\).

2 Preliminaries

The main subject of this work is the achievability of a certain task in a certain computational model. As we have already defined what a task is (Definition 1), we will now define the model - the postselection oracle algorithm - and what it means for a postselection oracle algorithm to achieve a task. We also review some existing algorithms.

2.1 Postselection oracle algorithm

In our definitions of oracle algorithms we follow the spirit of [BBC+01, BDW02], but in addition to the direct queries \(id(U) = U\) to the oracle \(U\), we allow for other types of queries, specified by functions on \(U(d)\), for example \(inv\) queries, i.e. queries to \(inv(U) = U^\dagger\).

**Definition 2 (Unitary oracle algorithm).** Let \(\Sigma\) be a collection of functions that map any \(U \in U(d)\) to a unitary. A unitary oracle algorithm making queries from \(\Sigma\) is a sequence of unitary transformations

\[V_0, \sigma_1(U), V_1, \sigma_2(U), \ldots, V_{N-1}, \sigma_N(U), V_N.\]

The string \(s = \sigma_1 \sigma_2 \ldots \sigma_N \in \Sigma^*\), called the query sequence, is fixed for the algorithm; the number of queries the algorithm makes is the length of the string \(|s| = N\) and the \(i\)-th letter of \(s\) determines the \(i\)-th query made by the algorithm, \(\sigma_i(U)\). Denote by \(\mathcal{H}_i\) the Hilbert space that \(\sigma_i(U)\) acts on, then the algorithm implements the operator

\[V_N(\sigma_N(U) \otimes 1_{K_N}) \ldots V_2(\sigma_2(U) \otimes 1_{K_2}) V_1(\sigma_1(U) \otimes 1_{K_1}) V_0 \in L(\mathcal{Z}),\]

to some Hilbert space \(\mathcal{Z}\), where \(1_{K_i} \in L(K_i)\) is the identity operator and the Hilbert space \(K_i\) is such that \(\mathcal{H}_i \otimes K_i = \mathcal{Z}\). Since it depends on the query, this decomposition of \(\mathcal{Z}\) to subsystems
may vary for different $i \in [N]$. The algorithm comes with two additional decompositions; $\mathcal{Z} = \mathcal{H} \otimes \mathcal{K}$ at the outset, and $\mathcal{Z} = \mathcal{H}' \otimes \mathcal{K}'$ at the end of the algorithm, where $\mathcal{K}$ and $\mathcal{K}'$ label the input and output ancilla registers.

We illustrate the fact that the query sequence is fixed for each algorithm by an example: Suppose that an algorithm is allowed to make calls to $U$ and $U^\dagger$, i.e. $\Sigma = \{id, inv\}$, and suppose that its query sequence is $s = \sigma_1\sigma_2 \ldots \sigma_N$ with $\sigma_i = inv$. The fact that $s$ is fixed means that at the $i$-th step the algorithm will always make a $U^\dagger$ query, independently of its current oracle $U$ and its current input.

**Definition 3 (Postselection oracle algorithm).** A postselection oracle algorithm making queries from $\Sigma$ and acting on the Hilbert space $\mathcal{Z}$ consists of

- a unitary oracle algorithm making queries from $\Sigma$, implementing an operator in $L(\mathcal{Z})$,
- a binary measurement $\{|\Pi, 1 - \Pi\rangle\}$, with $\Pi \in L(\mathcal{Z})$ a projector corresponding to ‘success’.

In other words, upon getting the ‘success’ outcome a postselection oracle algorithm implements $A(U) = \Pi V_N(\sigma_N(U) \otimes I_{K_N}) \ldots V_2(\sigma_2(U) \otimes I_{K_2}) V_1(\sigma_1(U) \otimes I_{K_1}) V_0,$ (1)

to the Hilbert space $\mathcal{Z}$. Moreover, the algorithm satisfies the following postselection condition: If $\mathcal{Z} = \mathcal{H} \otimes \mathcal{K}$ is the decomposition at the outset of the algorithm, i.e. $\mathcal{K}$ is the ancilla input register, then

$$||A(U)(|\xi\rangle \otimes |0\rangle_{\mathcal{K}})||^2 > 0$$

for all $U \in U(d)$ and for all $|\xi\rangle \in \mathcal{H}$. In other words, if the ancilla input Hilbert space is initialised to the all-zero state, then the probability of ‘success’ is always nonzero$^1$.

Figure 1 depicts a postselection oracle algorithm. Note that a unitary oracle algorithm is a postselection oracle algorithm with $\Pi = 1$. From now on we assume that the input and output decompositions of the total Hilbert space are the same: $\mathcal{H}' = \mathcal{H}$ and $\mathcal{K}' = \mathcal{K}$. Then we will often represent a postselection oracle algorithm by the function $A : U(d) \rightarrow L(\mathcal{H} \otimes \mathcal{K})$.

### 2.2 Achieving a task

We say that a postselection oracle algorithm $A$ achieves the task $(t, \Sigma)$ exactly, if it makes queries from $\Sigma$ and if, with the properly initialised ancilla, it implements $t(U)$ to $\mathcal{H}$. Formally:

**Definition 4 (Exactly achieving a task).** A postselection oracle algorithm $A : U(d) \rightarrow L(\mathcal{H}_A \otimes \mathcal{K})$ making queries from $\Sigma_A$ exactly achieves a task $(t : U(d) \rightarrow L(\mathcal{H}_t), \Sigma)$ if $\mathcal{H}_A = \mathcal{H}_t$, $\Sigma_A \subseteq \Sigma$ and if there exists some $U$-dependent ‘garbage’ operator $G_U \in L(\mathcal{K})$ such that the operator equation

$$A(U)(|1_{\mathcal{H}_t} \otimes |0\rangle_{\mathcal{K}}) = t(U) \otimes G_U$$

is satisfied.

$^1$Such a condition is standard in postselection models - see for example the definition of PostBQP in [Aar05]. Here the condition ensures our model is worst-case.
Equation (3) guarantees that algorithm $A$ implements $t(U)$ to the task register, if two conditions are met: The ancilla input to the algorithm is the all-zero state and the postselection measurement yields 'success'. These conditions are imposed by the $|0\rangle|0\rangle_K$ projector and by the presence of the $\Pi$ projector inside $A(U)$ (see eq. (1)). Moreover, observe that if eq. (3) holds for some unitary $t(U)$, then the algorithm’s postselection probability equals $||G_U|0\rangle||^2$.

2.3 Achieving a task when the query sequence is fixed

We remark that sometimes it is meaningful to first fix a query sequence $s \in \Sigma^*$ and then look for an algorithm associated with this query sequence which achieves a specific task. One is then effectively looking only for the unitaries $\{V_i\}_{0 \leq i \leq |s|}$ and the projector $\Pi$. This is the case in the impossibility due to Araújo et al. [AFCB14], where $|s| = 1$, $s = id$, or in the neutralisation task we will discuss in Definition 6.

2.4 Properties of postselection oracle algorithms

Since any postselection oracle algorithm is built as a product of operators of eq. (1), the following properties are immediate:

**Property 1 (Continuity).** Let $\Sigma$ be a set of continuous functions. Any postselection oracle algorithm $A : U(d) \rightarrow L(H \otimes K)$ making queries from $\Sigma$ must be a continuous function.

For example, any algorithm for a task with $\Sigma = \{id, inv\}$ must be continuous, while there may exist a discontinuous algorithm for a task with $\Sigma = \{U \mapsto U^q\}_{q \in \mathbb{Q}}$.

Another property that a postselection oracle algorithm inherits from the letters of its query alphabet is homogeneity:

**Definition 5.** A function $h : U(d) \rightarrow L(H)$ is called $m$-homogeneous if for every $U \in U(d)$ and $\lambda \in S^1$ we have $h(\lambda U) = \lambda^m h(U)$.

**Property 2 (Homogeneity).** Let $\Sigma$ be a set of homogeneous functions. Denote by $m_\sigma$ the degree of homogeneity of the function $\sigma \in \Sigma$. Any postselection oracle algorithm $A : U(d) \rightarrow L(H \otimes K)$ making queries from $\Sigma$ must be $(\sum_{\sigma \in \Sigma} \#_\sigma(s) m_\sigma)$-homogeneous, where $\#_\sigma(s)$ is the number of occurrences of the letter $\sigma$ in $A$’s query sequence $s$.

For example, any algorithm making queries from $\{id\}$ is $|s|$-homogeneous, and an algorithm making queries from $\{id, inv\}$ is $\Delta$-homogeneous, with $\Delta = \#_{id}(s) - \#_{inv}(s)$.

2.5 Review of the controlled-swap subroutine and its use in the $q$-th power task

The controlled-swap algorithm of Kitaev [Kit95] makes one call to $U \in U(d)$ and implements $A^1_t(U) = |0\rangle|0\rangle \otimes 1 \otimes U_{K_2} + |1\rangle|1\rangle \otimes U \otimes 1_{K_2}$ using controlled-swap gates (see Figure 2). It implements control$_\sigma(U)$ only if the last qudit $K_1$ is considered an ancilla and is initialised to an eigenvector of $U$. Since the eigenvector of $U$ is $U$-dependent, $A^1_t : U(d) \rightarrow L(H \otimes K_2)$ does not exactly achieve $c\cdot U$.

However, in some applications $A^1_t$ can replace control queries. An example is the average-case $q$-th power algorithm of Sheridan et al. [SMM09]. The first version of their algorithm makes control and inv$\circ$control queries and implements an operator which, for most $U \in U(d)$, is close to $U^q \otimes 1_{K_2}$. How close it is, depends on the dimension of the ancilla Hilbert space $K_q$, to which it (approximately) implements the garbage operator $1_{K_q}$ (See also Gilányi et al. [GSLW19] for an alternative algorithm which assumes control queries). In Appendix A of [SMM09] Sheridan et al. modify their algorithm to the version we are interested in - corresponding to the task $(U \mapsto U^q, \{id, inv\})$: They replace the control and inv$\circ$control queries by $A^1_t(U)$ and $A^1_t(U)^\dagger$. 

6
subroutines, adding a qudit register $K_\uparrow$ to the ancilla. This version implements an operator close to $U^q \otimes U_\uparrow^{-q} \otimes 1_{K_\downarrow}$ for most $U \in U(d)$, the garbage operator now being $U$-dependent. Indeed, since this second version uses only $id$ and $inv$ queries, the total implemented operator must be insensitive to the global phase of the oracle $U$.

2.6 Review of the c-$U^d$ algorithm via neutralisation

In this section we review the c-$U^d$ algorithm of Dong et al. [DNSM19]. Suppose we want to check for what $m \in \mathbb{Z}$ we can exactly achieve c-$U^m$ taking the following approach: (i) query $U$ in sequence $m$ times if the control is $|1\rangle$ and if the control is $|0\rangle$, 'move' the $m$ queries to an ancilla register via controlled-swap gates as in Figure 2 and (ii) if the control is $|0\rangle$, run a subroutine on the ancilla which makes the 'moved' queries have no effect other than giving its input a phase (and possibly multiplying its norm by a constant). Dong et al. say that the subroutine of step (ii) neutralizes the 'moved' queries. Fitting neutralisation into our framework we define:

**Definition 6 (Neutralising a query sequence).** Given a query sequence $s$, a postselection oracle algorithm $A_\emptyset : U(d) \rightarrow L(\mathbb{Z})$ neutralises $s$, if its query sequence is $s$ and if the following operator equation

$$A_\emptyset(U) |0\rangle\langle 0| = r e^{i\phi(U)} |0\rangle\langle 0|$$

is satisfied, where $|0\rangle\langle 0| \in L(\mathbb{Z})$ is the projection of the entire Hilbert space onto the all-zero vector, $\phi(U)$ is some real function of $U$, and $r \in (0,1]$.

Observe that this exactly corresponds to our requirement of the subroutine in step (ii). If such an $A_\emptyset$ exists, then we have an algorithm exactly achieving c-$U^m$ (it only remains to add rescaling by $r$ to the control-$|1\rangle$ case). Dong et al. [DNSM19] noted that Shur-Weyl duality implies that parallel $id$ queries can be neutralised if and only if their number is a multiple of the oracle’s dimension $d$. They presented a unitary oracle algorithm neutralising $s = (id)^d$ and used it to build the algorithm of Figure 3.

**Construction 1 (due to [DNSM19]).** A unitary $V$ exists such that the unitary oracle algorithm of Figure 3 exactly achieves the task c-$U^d = (U \mapsto \text{control}_\emptyset(U^d), \{id, inv\})$.

In particular, their algorithm $A : U(d) \rightarrow L(\mathcal{H}_\uparrow \otimes \mathcal{K})$ makes $d$ $id$ queries, and if its $d$ ancilla qudits are all initialised to zero, it implements

$$A(U) \left( 1_{\mathcal{H}_\uparrow} \otimes |0\rangle\langle 0|_\mathcal{K} \right) = \left( |0\rangle\langle 0| \otimes 1 + \frac{1}{\det(U)} |1\rangle\langle 1| \otimes U^d \right) \otimes \left( \det(U) |0\rangle\langle 0|_\mathcal{K} \right).$$

Figure 2: The controlled-swap subroutine $A_\uparrow$ of [Kit95].
Figure 3: The unitary oracle algorithm for the $c$-$U^d$ task due to [DNSM19].

Proof. First note that we can write the determinant as

$$
\det (U) = \sum_{\pi \in S_d} \sgn (\pi) \prod_{i=1}^{d} \langle i | U | \pi(i) \rangle
$$

$$
= \frac{1}{d!} \sum_{\pi, \tau \in S_d} \sgn (\tau) \sgn (\pi) \prod_{i=1}^{d} \langle \tau(i) | U | \pi(i) \rangle
$$

(5)

$$
= \langle \chi_d | U \otimes | \chi_d \rangle,
$$

where $| \chi_d \rangle = \frac{1}{\sqrt{d}} \sum_{\pi \in S_d} \sgn (\pi) | \pi(1) \rangle \otimes \cdots \otimes | \pi(d) \rangle \in \mathcal{K}$. For a proof of eq. (5) see Appendix C. Observe that $| \chi_d \rangle$ is normalised and therefore there exists a unitary $V \in \mathcal{L}(\mathcal{K})$ such that $V | 0 \rangle = | \chi_d \rangle$. Define the unitary oracle algorithm $A_\phi(U) := V^\dagger U \otimes | \chi_d \rangle$. Since $| \langle \chi_d | U \otimes | \chi_d \rangle | = 1$, the unitary $A_\phi(U)$ on input $| 0 \rangle$ outputs $| 0 \rangle$ with probability 1 for all $U \in U(d)$. Therefore $A_\phi$ neutralises its queries with $r = 1$, the corresponding equation being $A_\phi(U) | 0 \rangle | 0 \rangle = \det(U) | 0 \rangle | 0 \rangle$. Observe that the algorithm of Figure 3 implements $| 0 \rangle | 0 \rangle \otimes \mathbb{I} \otimes A_\phi(U) \mathbb{I} + | 1 \rangle | 1 \rangle \otimes U^d \otimes \mathbb{I} \mathbb{I}$, from which eq. (4) follows.

3 The Exact Dichotomy

Now we are ready to prove the dichotomy regarding the $c$-$U^m$ task. In this section we will prove the exact version:

**Theorem 1 (The Exact Dichotomy).** Let $m \in \mathbb{Z}$ and let $d \in \mathbb{N}$ be the dimension of the oracle, $U \in U(d)$.

- If $d | m$ there exists a postselection oracle algorithm exactly achieving the task
  
  $$
  c-U^m := (U \mapsto | 0 \rangle | 0 \rangle \otimes \mathbb{I} + e^{i\phi(U)} | 1 \rangle | 1 \rangle \otimes U^m, \{id, \text{inv}\}),
  $$

  for some $\phi : U(d) \to \mathbb{R}$.

- If $d \nmid m$ no such algorithm exists.

The existence in Theorem 1 is satisfied by the $c$-$U^d$ algorithm of [DNSM19]:

**Proof of Theorem 1 ($d | m$ case).** For positive $m$ which is an integer multiple of $d$, one can simply apply [DNSM19] algorithm (see Figure 3) $\frac{m}{d}$ times, for $m$ negative also replace the oracle calls to $U$ by oracle calls to $U^\dagger$. □
We prove the $d \nmid m$ case or the impossibility direction of Theorem 1 here, and then de facto again in Section 4 in the approximate dichotomy. In each proof, we use a different way to reduce a $c\cdot U^m$ algorithm to a continuous $m$-homogeneous function from $U(d)$ to the circle $S^1$, after which we apply the following topological lemma:

**Lemma 1.** Let $d \in \mathbb{N}$ and $m \in \mathbb{Z}$ such that there exists a continuous $m$-homogeneous function $f : U(d) \rightarrow S^1$. Then $m$ is a multiple of $d$.

The following reduction has an operational meaning; the function $U(d) \rightarrow S^1$ we will build from the assumed $c\cdot U^m$ algorithm is itself a postselection oracle algorithm (implementing a 1-dimensional operator). For the approximate dichotomy of Section 4, we will use a reduction with no operational meaning for the sake of obtaining a stronger error bound.

**Proof of Theorem 1 ($d \nmid m$ case).** Let $m \in \mathbb{Z}$ and let $A : U(d) \rightarrow L(\mathcal{H} \otimes \mathcal{K})$ be a postselection oracle algorithm exactly achieving $c\cdot U^m$. We will show that $d \mid m$.

Any postselection oracle algorithm making queries from $\Sigma = \{id, inv\}$

- is by Property 1 continuous,
- is by Property 2 $\Delta$-homogeneous for some $\Delta \in \mathbb{Z}$.

Define the functions

$$h(U) := \langle 0 | p(U(U^m \otimes 1) | 0 \rangle \left( | 1 \rangle \langle A(U) | 1 \rangle \right) \langle 0 | A(U) | 0 \rangle \langle 0 | \rangle,$$

$$p_0(U) := \langle 0 | A(U)^\dagger A(U) | 0 \rangle,$$

where $| 0 \rangle$ is the all-zero state over all registers, $| 0 \rangle \_c$ and $| 1 \rangle \_c$ are the computational basis states of the first qubit and $| 0 \rangle \_c$ the all-zero state on what is left (i.e. $| 0 \rangle \_c \otimes | 0 \rangle \_c = | 0 \rangle$). From the listed properties of $A$ it follows that $h$ is continuous and $(m - \Delta + \Delta = m)$-homogeneous (since $U \mapsto A(U)^\dagger$ is $(-\Delta)$-homogeneous), and $p_0$ is continuous and 0-homogeneous. Note that $p_0$ is the postselection probability of $A(U)$ applied to the all-zero ancilla and task input, $| 0 \rangle$.

Next we use the fact that $A$ exactly achieves the $c\cdot U^m$ task, i.e. there is a real function $\phi$ and a $U$-dependent ancilla operator $G_U \in L(\mathcal{K})$ such that

$$A(U) (1 \otimes | 0 \rangle \_c) = t(U) \otimes G_U = \left( | 0 \rangle \langle 0 \| \_c \otimes 1 + e^{i\phi(U)} | 1 \rangle \langle 1 \| \_c \otimes U^m \right) \otimes G_U.$$

Substituting this into $p_0$, the unitarity of $t(U)$ gives $p_0(U) = \langle 0 | \_c G_U^\dagger G_U | 0 \rangle \_c$. Substituting eq. (6) into $h$, we obtain

$$h(U) = \langle 0 | p(U(U^m \otimes 1) | 0 \rangle \left( | 1 \rangle \langle A(U) | 1 \rangle \right) \langle 0 | A(U) | 0 \rangle \langle 0 | \rangle,$$

$$= e^{-i\phi(U)} \langle 0 | \_c G_U^\dagger G_U | 0 \rangle \_c.$$

Because of the postselection condition $p_0(U) = | h(U) |$ is never zero and the function $f(U) := h(U)/|h(U)|$ is well-defined. Moreover, it is continuous $m$-homogeneous and extracts the relative phase, i.e. $f : U(d) \rightarrow S^1$. Then from Lemma 1 we deduce that $m$ is a multiple of $d$. □

Note that in the above we refer to the specific task function $t(U) = control_\phi(U^m)$ quite late in the proof. This is to emphasise the versatility of our method: algorithms achieving other tasks may as well reduce to continuous homogeneous functions between spaces with a nontrivial fundamental group (which, as we show next, is the key to Lemma 1). See the direct proof of Corollary 2 for another application of this method.

Our proof of Lemma 1 uses the algebraic topology machinery of fundamental groups. The background needed to understand the proof, all of which is covered in Chapter 9 of [Mun00], is just the basics of fundamental groups (paths, homotopies, time reparametrization etc.) together with the description of the fundamental group of the circle $S^1$ as infinite cyclic and
generated by the homotopy class of the loop $t \mapsto e^{2\pi it}$. Throughout the proof, we will use the notation $\pi_1 (X, p)$ for the fundamental group of the topological space $X$ based at $p$. For a path $\gamma : [0, 1] \to X$, we will denote by $[\gamma]$ its homotopy class, and for another path $\gamma'$ such that $\gamma (1) = \gamma' (0)$ we will denote by $\gamma * \gamma'$ their composition.

**Lemma 2.** Let $\gamma : [0, 1] \to U (d)$ be the loop $\gamma (t) = e^{2\pi it} \mathbb{I}$, where $\mathbb{I} \in U (d)$ is the identity matrix. Then its homotopy class $[\gamma]$ is a $d$-th power in the fundamental group $\pi_1 (U (d), \mathbb{I})$, i.e. there exists some other loop $\gamma' : [0, 1] \to U (d)$ based at $\mathbb{I}$ such that $[\gamma']^d = [\gamma]$.

We provide two proofs: the first is shorter whereas the second is self-contained.

**First proof.** We use the fact (see Proposition 2.2.6 of [MS17]) that the determinant map $\det : U (d) \to S^1$ induces an isomorphism on the corresponding fundamental groups $\det_* : \pi_1 (U (d), \mathbb{I}) \to \pi_1 (S^1, 1)$. The fundamental group $\pi_1 (S^1, 1)$ is isomorphic to $\mathbb{Z}$ and generated by the homotopy class of the loop $\nu : [0, 1] \to S^1$ defined by $\nu (t) = e^{2\pi it}$. Since $\text{det} \circ \gamma : [0, 1] \to S^1$ is the loop $t \mapsto e^{2\pi it d}$ which winds $d$ times around the circle, it is homotopic to the composition of $\nu$ with itself $d$ times (as both loops are equal up to reparametrization of time). Thus,

$$\text{det}_* ([\gamma]) = [\text{det} \circ \gamma] = \left[ \nu \ast \nu \ast \ldots \ast \nu \right]_{\text{d times}} = [\nu] \cdot [\nu] \cdot \ldots \cdot [\nu] = [\nu]^d.$$  

(7)

Since $\text{det}_* ([\gamma])$ is a $d$-th power in $\pi_1 (S^1, 1)$ and $\text{det}_*$ is a group isomorphism, we deduce that $[\gamma]$ is also a $d$-th power (to see this, apply the inverse of $\text{det}_*$ to both sides of eq. (7)).

**Second proof.** We will give an explicit element in $\pi_1 (U (d), \mathbb{I})$ such that its $d$-th power is $[\gamma]$. For every $1 \leq j \leq d$, set $\gamma_j (t)$ to be the diagonal matrix having 1 along the diagonal except the $j$-th entry which is $e^{2\pi it}$. We first note that the loops $\{\gamma_j\}_{j=1}^d$ are all homotopic: an explicit homotopy between $\gamma_1$ and $\gamma_j$ is given by

$$H_j (s, t) = R_j (s) \gamma_1 (t) R_j (s)^\dagger,$$

where

$$R_j (s) = \begin{pmatrix}
\cos \left( \frac{\pi}{2} s \right) & i \sin \left( \frac{\pi}{2} s \right) \\
1 & 1 \\
\vdots & 1
\end{pmatrix}.$$ 

So we can now deduce that $[\gamma_1]^d = [\gamma_1] \cdot [\gamma_2] \cdot \ldots \cdot [\gamma_d] = [\gamma_1 * \gamma_2 * \ldots * \gamma_d]$. It remains to show that $\gamma_1 * \gamma_2 * \ldots * \gamma_d$ is homotopic to $\gamma$. To that end, let $f, g : [0, 1] \to [0, 1]^d$ be defined by $f (t) = (t, t, \ldots, t)$ and

$$g (t) = \begin{cases}
(t d, 0, 0, \ldots, 0), & t \leq \frac{1}{d} \\
(1, td - 1, 0, \ldots, 0), & \frac{1}{d} \leq t \leq \frac{2}{d}, \\
\vdots \\
(1, 1, \ldots, 1, td - (d - 1)), & \frac{d-1}{d} \leq t \leq 1
\end{cases}.$$ 

Note that both $f$ and $g$ are paths inside the $d$-dimensional cube from $(0, 0, \ldots, 0)$ to $(1, 1, \ldots, 1)$, and while $f$ moves linearly along the cube’s diagonal, $g$ takes a path along the sides of the cube.
Since the cube $[0,1]^d$ is convex, it is simply connected and so $f$ and $g$ are homotopic paths. Define $\varphi : [0,1]^d \to U(d)$ by

$$\varphi(x_1,x_2,\ldots,x_d) = \begin{pmatrix}
  e^{2\pi i x_1} \\
  e^{2\pi i x_2} \\
  \vdots \\
  e^{2\pi i x_d}
\end{pmatrix},$$

and note that $\varphi \circ f$ and $\varphi \circ g$ are homotopic (a homotopy between them can be obtained from a homotopy between $f$ and $g$ by composing with $\varphi$). Observing that $\varphi \circ f = \gamma$ and that $\varphi \circ g$ is equal to $\gamma_1 \ast \gamma_2 \ast \ldots \ast \gamma_d$ up to reparametrization of time, we finish the proof as

$$[\gamma] = [\varphi \circ f] = [\varphi \circ g] = [\gamma_1 \ast \gamma_2 \ast \ldots \ast \gamma_d] = [\gamma_1]^d.$$

\[\square\]

\textbf{Proof of Lemma 1.} Without the loss of generality we may assume that $h(1) = 1$, since otherwise we may replace $h$ with its rotation $U \mapsto h(1)^{-1}h(U)$. Let $\gamma : [0,1] \to U(d)$ be the loop $\gamma(t) = e^{2\pi i t}1$. By Lemma 2, the homotopy class $[\gamma]$ is a $d$-th power in $\pi_1(U(d),1)$, and so must map to a $d$-th power under any group homomorphism. In particular, $h_*([\gamma]) = [h \circ \gamma]$ is a $d$-th power in $\pi_1(S^1,1)$, and so we can write $[h \circ \gamma] = [\gamma']^d$ for some loop $\gamma' : [0,1] \to S^1$. Since the fundamental group of the circle $\pi_1(S^1,1)$ is (infinite) cyclic and generated by the homotopy class of the loop $\nu(t) = e^{2\pi i t}$, we can write $[\gamma'] = [\nu]^k$ for some $k \in \mathbb{Z}$, and so we have $[h \circ \gamma] = [\nu]^kd$. On the other hand, the $m$-homogeneity of $h$ allows us to derive an explicit formula for $h \circ \gamma$:

$$h \circ \gamma(t) = h(e^{2\pi i t}1) = (e^{2\pi i t})^m \cdot h(1) = e^{2\pi imt}.$$

The loop $t \mapsto e^{2\pi imt}$, which winds $m$ times around the circle, is homotopic to the composition of $\nu$ with itself $m$ times (as both loops are equal up to reparametrization of time) and so $[h \circ \gamma] = [\nu]^m$. Combining the two expressions we got for $[h \circ \gamma]$ in terms of $[\nu]$, we deduce that $[\nu]^{kd} = [\nu]^m$. Since the order of $[\nu]$ in $\pi_1(S^1,1)$ is infinite, the two powers must be equal and $m = kd$.\[\square\]

\subsection{Consequences for neutralisation}

Dong et al. [DNSM19] observed that parallel \textit{id} queries cannot be neutralised if their number is not a multiple of the oracle’s dimension. Here we show that this remains impossible also when the parallel assumption is dropped. Our first proof, a reduction to Theorem 1, is similar to their construction of $c-U^d$ algorithm (Section 2.6).

\textbf{Corollary 2.} Let $d \in \mathbb{N}$ be the dimension of the oracle, $U \in U(d)$, and $\text{id}$ represent a query sequence $\text{id} = \text{id}\text{id} \ldots \text{id}$, where $\text{id}(U) = U$. If $d \nmid |\text{id}|$ then there is no neutralising algorithm for this query sequence.

\textbf{Proof 1 (via Theorem 1).} Let $A_\emptyset$ be a neutralising algorithm for $\text{id}$, with the corresponding constant $r \in (0,1]$ (see Definition 6). Construct a new algorithm which runs $A_\emptyset$ if a control qubit is $|0\rangle$ and makes the $|\text{id}|$ oracle calls in sequence if the control is $|1\rangle$. If $r \neq 1$, lower the probability of postselection in the control-$|1\rangle$ case to match $r$ for example by adding a tuned $|1\rangle$-controlled rotation of the ancilla state before the final projection of the ancilla. We obtain an algorithm exactly achieving $c-U^{|\text{id}|}$. By Theorem 1 $|\text{id}|$ is a multiple of $d$.\[\square\]
Proof 2 (direct). Assume \( A_\varnothing \) is a neutralising algorithm for \( \text{id} \). This \( A_\varnothing : U(d) \rightarrow L(\mathcal{Z}) \) is by Property 1 continuous and by Property 2 \( |\text{id}| \)-homogeneous and so is

\[
|\langle 0 | A_\varnothing(U) | 0 \rangle| = |\langle 0 | A_\varnothing(U)^\dagger A_\varnothing(U) | 0 \rangle| = e^{i\phi(U)},
\]

which maps from \( U(d) \) to the circle. Then by Lemma 1, \( |\text{id}| \) is a multiple of \( d \).

3.2 Consequences for the transpose and inversion tasks

In this section we show how Theorem 1 precludes transpose and inversion algorithms with certain numbers of oracle calls from being clean. A clean algorithm implements (up to a global phase) a constant operator to the ancilla. Formally,

Definition 7. [Clean postselection oracle algorithm]

Let \( A : U(d) \rightarrow L(\mathcal{H}_t \otimes K) \) be a postselection oracle algorithm that exactly achieves the task \( (t : U(d) \rightarrow L(\mathcal{H}_t), \Sigma) \), i.e. \( A(U)(1 \otimes |0\rangle) = t(U) \otimes G_U \) for some \( G_U \in L(\mathcal{K}) \). We call \( A \) clean if we can write the garbage operation as \( G_U = e^{ig(U)}G \) for some constant \( G \in L(\mathcal{K}) \) and some \( g : U(d) \rightarrow \mathbb{R} \).

Cleanness might be useful if the algorithm is to be composed coherently as a subroutine inside a larger algorithm. A subroutine with a \( U \)-dependent ancilla operator \( G_U \) would prevent interference (since \( U \) is unknown).

For every \( k \in \mathbb{N} \), Quintino et al. [QDS19a, QDS19b] presented a \( k \)-query postselection oracle algorithm exactly achieving the transpose task \( (U \rightarrow U^T, \{\text{id}\}) \) and a \( (k(d-1)) \)-query postselection oracle algorithm exactly achieving the inverse task \( (\text{inv}, \{\text{id}\}) \). The success probability of these grows with \( k \). As we show next, they can be clean only at specific \( k \)-values. (See Appendix B for a review of the clean \( k = 1 \) algorithms.)

Corollary 3. If there is a clean algorithm making \( N \) oracle calls and exactly achieving

1. the inverse task \( (t : U \rightarrow U^\dagger, \Sigma = \{\text{id}\}) \), then \( N = -1 \mod d \).
2. the transpose task \( (t : U \rightarrow U^T, \Sigma = \{\text{id}\}) \), then \( N = 1 \mod d \).

Proof of 1. If a clean algorithm with \( N \) calls to \( \text{id}(U) \) implements \( U^\dagger \) on \( \mathcal{H}_t \), then with another \( \text{id} \) query implemented to \( \mathcal{H}_t \) it is a clean algorithm making \( N + 1 \) calls and implementing the identity, i.e. it neutralises \( N + 1 \) calls. Formally, call \( A_{\text{inv}} : U(d) \rightarrow L(\mathcal{H}_t \otimes K) \) the clean \( N \)-query algorithm exactly achieving the inverse task, \( A_{\text{inv}}(U)(1 \otimes |0\rangle) = e^{ig(U)}U^\dagger \otimes G \) as in Definition 7. Then

\[
(U \otimes |0\rangle)A_{\text{inv}}(1)^\dagger A_{\text{inv}}(U)(1 \otimes |0\rangle) = r e^{i(g(U)-g(1))} 1 \otimes |0\rangle|0\rangle
\]

with \( r = \langle 0 | G^1G | 0 \rangle \), is a neutralisation algorithm for the string \( (\text{id})^N+1 \) (since eq. (2) holds for \( A_{\text{inv}} \), \( r \) is nonzero). Corollary 2 tells us that \( d \) must divide \( N + 1 \), therefore \( N = -1 \mod d \).

Proof of 2. Assume there is a clean algorithm making \( k \) calls and exactly achieving the transpose task. Compose it with the clean unitary oracle algorithm for complex conjugation by Miyazaki, Soeda and Murao [MSM19] (reviewed in Appendix B.1) to obtain clean inversion algorithm making \( k(d-1) \) oracle calls. Then by the above argument \( k(d-1) + 1 \) is a multiple of \( d \) and it follows that \( k = 1 \mod d \).

4 The Approximate Dichotomy

In this section we show that the dichotomy of Theorem 1 holds also if we ask only for an approximation of the desired task operator. To state this precisely, we need to replace the notion of exactly achieving a task (Definition 4) by a notion that allows for some error.
4.1 Approximating a task

Here we define a weaker version of Definition 4. So far we have used pure states formalism and represented computations by operators on pure states. To be able to discuss approximations, we now switch to the density matrix formalism. In the following, we denote by $\mathcal{D}(\mathcal{H}) \subset L(\mathcal{H})$ the set of density operators on $\mathcal{H}$.

**Definition 8 ($\epsilon$-approximating a task).**

A postselection oracle algorithm $A : U(d) \to L(\mathcal{H}_A \otimes \mathcal{K})$ making queries from $\Sigma_A$ $\epsilon$-approximates the task $(t : U(d) \to L(\mathcal{H}_t), \Sigma)$ if $\mathcal{H}_A = \mathcal{H}_t$, $\Sigma_A \subseteq \Sigma$ and if for all $U \in U(d)$, and for all task input states $\rho \in \mathcal{D}(\mathcal{H}_t)$ the renormalised output on the task register is close to the task output, i.e.

$$\sup_{\rho \in \mathcal{D}(\mathcal{H}_t)} \left\| \frac{A_U(\rho)}{tr[A_U(\rho)]]} - t(U)\rho t(U)\right\|_{tr} \leq \epsilon,$$

(8)

where $\| \cdot \|_{tr}$ is the trace norm, and the superoperator $A_U : L(\mathcal{H}_t) \to L(\mathcal{H}_t)$ defined as $A_U(\rho) = tr_K [A(U)(\rho \otimes |0\rangle\langle 0|_K)] A(U)^\dagger$ captures how the postselection oracle algorithm acts on the task register.

Note that in this formalism the postselection condition of eq. (2) can be reformulated as $tr[A_U(\rho)] > 0$ for all $\rho \in \mathcal{D}(\mathcal{H}_t)$ and $U \in U(d)$, therefore inequality (8) is well-defined.

We note that there is a subtlety in the left-hand side of inequality (8) - the output of $A_U$ is renormalised, making the expression inside the trace norm nonlinear in $\rho \in \mathcal{D}(\mathcal{H})$. This is unavoidable, given the fact that we are handling postselection.

Observe furthermore, that the left-hand side of (8) is reminiscent of the trace induced distance of two superoperators (except for the difference of the renormalization). We could have also worked with a similar modification of the diamond distance, studied in [Gav20], which would correspond to taking the supremum in (8) over a larger set. We prefer not to do so, since first, this would have made the discussion more cumbersome technically, and second, our main result is an impossibility of certain upper bounds, namely of all $\epsilon < \frac{1}{2}$, for a certain task. In other words, we show that for this task the supremum in (8) must be large, which is a stronger claim when the supremum is over the smaller set.

By the next claim Definitions 4 and 8 are consistent.

**Claim 1.** Let the task $(t : U(d) \to L(\mathcal{H}_t), \Sigma)$ be such that $t(U)$ is unitary for all $U \in U(d)$. Then a postselection oracle algorithm exactly achieves $(t, \Sigma)$ if and only if it $0$-approximates $(t, \Sigma)$.

**Proof.** For the if direction, substitute eq. (3) into $A_U(\rho)$. For the only if assume $A$ $\epsilon$-approximates $(t, \Sigma)$. By Theorem 2.3 of [Gav20] for all $U \in U(d)$ there exists $|g_U\rangle \in \mathcal{K}$, such that

$$\|A(U)(1_{\mathcal{H}_t} \otimes |0\rangle\langle 0|_K) - t(U) \otimes |g_U\rangle\rangle_{op} \leq O(\sqrt{\epsilon}),$$

(9)

where $\| \cdot \|_{op}$ is the operator norm. Equation (3) follows by substituting $\epsilon = 0$ and taking $G_U = |g_U\rangle\langle 0|_K$. \[ \]

4.2 The Approximate Dichotomy Theorem

In this section we extend the exact dichotomy of Theorem 1 to errors as large as $\epsilon < \frac{1}{2}$. To obtain the bound of $\frac{1}{2}$ we use a new reduction to Lemma 1 which, as opposed to the reduction in Section 3, does not have a simple operational meaning.
Theorem 2 (The Approximate Dichotomy). Let \( m \in \mathbb{Z} \) and let \( d \in \mathbb{N} \) be the dimension of the oracle, \( U \in U(d) \). Let \( \epsilon \in [0, \frac{1}{2}] \).

- If \( d \mid m \) there exists a postselection oracle algorithm \( \epsilon \)-approximating the task
  \[
  c U^m := (U \mapsto |0\rangle\langle 0| \otimes 1 + e^{i\phi(U)} |1\rangle \otimes U^m, \{id, inv\}),
  \]
  for some \( \phi : U(d) \to \mathbb{R} \).
- If \( d \nmid m \) no such postselection oracle algorithm exists.

In particular, we obtain an impossibility of an approximate \( c U \):

Corollary 4 (Impossibility of Approximating Controlled \( U \)). In the postselection oracle algorithm model \( \epsilon \)-approximating the task \( c U \) is impossible for any \( \epsilon \in [0, \frac{1}{2}] \).

Proof of Theorem 2. The \( d \mid m \) direction follows from the fact that the algorithm of [DNSM19] exactly achieves \( c U^d \) and therefore also \( \epsilon \)-approximates it for any \( \epsilon \). For the \( d \nmid m \) direction, assume there exists a postselection oracle algorithm, \( A : U(d) \to L(H \otimes K) \), \( \epsilon \)-approximating \( c U^m \) for some \( \epsilon < \frac{1}{2} \). We will show that Lemma 1 again implies that \( m \) must be a multiple of \( d \).

Define the following functions \( U(d) \to \mathbb{C} \):

-  
  \[
  h_+(U) := (0)_c \langle + |_c A(U)^\dagger | 1\rangle_c (U^m \otimes 1_K)(0)_c A(U) | _+ \rangle_c | 0\rangle_\varphi
  \]
  \[
  p_+(U) := \langle + |_c \otimes (0)_\varphi A(U)^\dagger A(U) \langle _+ |_c \otimes | 0\rangle_\varphi
  \]
  \[
  f_+(U) := \frac{h_+(U)}{p_+(U)}
  \]

which are modifications of the functions defined in the proof of Theorem 1. In particular, the subscript \( c \) again denotes the first qubit, and the subscript \( \varphi \) the remaining qubits of all the registers (so that \( |0\rangle_c \otimes |0\rangle_\varphi \in H \otimes K \)). Note again that \( h_+ \) and \( f_+ \) are continuous and \( m \)-homogeneous and \( f_+ \) is well defined since \( p_+(U) = \text{tr} [A_U(\rho^*)] \) is the postselection probability on the task input \( \rho^* = |+\rangle c \otimes |0\rangle 0 \in L(H_c) \) and is therefore never zero. Moreover note that

-  
  \[
  \text{tr} [A_U(\rho^*) (|1\rangle_c \otimes U^m)] = h_+(U)
  \]
  \[
  \text{tr} [t(U) \rho^* t(U)^\dagger (|1\rangle_c \otimes U^m)] = \frac{1}{2} e^{-i\phi(U)},
  \]

for \( t(U) = |0\rangle\langle 0| \otimes 1 + e^{i\phi(U)} |1\rangle \otimes U^m \). We get

-  
  \[
  f_+(U) - \frac{1}{2} e^{-i\phi(U)} = \frac{\text{tr} [A_U(\rho^*) (|1\rangle_c \otimes U^m)]}{p_+(U)} - \text{tr} [t(U) \rho^* t(U)^\dagger (|1\rangle_c \otimes U^m)]
  \]
  \[
  \leq \left| \frac{A_U(\rho^*) (|1\rangle_c \otimes U^m)}{p_+(U)} - t(U) \rho^* t(U)^\dagger (|1\rangle_c \otimes U^m) \right|_{tr}
  \]
  \[
  \leq \left| \frac{A_U(\rho^*) - t(U) \rho^* t(U)^\dagger}{\text{tr} [A_U(\rho^*)]} \right| \left| |1\rangle_c \otimes U^m \right|_{op} \leq \epsilon
  \]

where \( \| \cdot \|_{op} \) is the operator norm. The first and the second inequality are applications of the general inequalities \( \text{tr}(X) \leq \|X\|_{tr} \) and \( \|XY\|_{tr} \leq \|X\|_{tr} \|Y\|_{op} \) which hold for any operators \( X, Y \in L(H_c) \) (see Lemma 10 of [AKN98]) and the last is the assumption that the algorithm \( \epsilon \)-approximates \( c U^m \). We got that

-  
  \[
  |f_+(U)| \geq \frac{1}{2} - \epsilon > 0,
  \]

for all \( U \in U(d) \) since we had \( \epsilon < \frac{1}{2} \). This lets us define \( f(U) := f_+(U)/|f_+(U)| \) which is a continuous \( m \)-homogeneous function \( U(d) \to S^1 \). By Lemma 1, \( m \) is a multiple of \( d \). \( \square \)
5 Worst-case versus average-case

We have noted before that our model is worst case. Specifically, we require that the task operator is (approximately) implemented whenever the algorithm outputs ‘success’ and, at the same time, that the probability of ‘success’ is always nonvanishing. Therefore, the task operator \( t(U) \) is (approximately) implemented with nonzero probability for all oracles \( U \in U(d) \).

In this section we compare our impossibility result to the algorithms of [SMM09] and [DNSM19] which, when combined, give an average-case \( c-U \) algorithm. First we show an impossibility of a worst-case version of [SMM09], then we ask about variations of the [DNSM19] algorithm, which would render the composed average-case \( c-U \) algorithm efficient.

5.1 A fractional power task

We use the example of the \( \frac{1}{d} \)-th power task to emphasise that our model is worst-case. Approximately implementing \( U^\frac{1}{d} \) within any error \( \epsilon > 0 \) can be done by the average-case algorithm of [SMM09] (\( \epsilon \) determines the algorithm’s complexity), while we prove next that for a small enough \( \epsilon \) this becomes impossible within our model. The algorithm of [SMM09] is outside our model because it fails for some \( U \in U(d) \). Their Lemma 1 implies that this is necessary. So does the following corollary of Theorem 2, but without any reference to the algorithm’s oracle complexity.

**Corollary 5.** Let the \( \frac{1}{d} \)-th power task be \( (t, \{\text{id, inv}\}) \), where the function
\[
t : U(d) \to U(d) \\
t(U) = U^\frac{1}{d}
\]
for each \( U \in U(d) \) outputs a specific \( d \)-th root of \( U \) (as there could be several). If some postselection oracle algorithm \( \epsilon \)-approximates this task, then \( \epsilon = \Omega(d^{-2}) \).

We remark that the lower bound on \( \epsilon \) scales as \( d^{-2} \) and not \( d^{-1} \) due to our relatively weak notion of \( \epsilon \)-approximating, (see the discussion under Definition 8). Note that intuitively, the worst-case impossibility is not surprising: we are trying to approximate the discontinuous function \( t \) by some continuous postselection oracle algorithm.

**Proof.** Assume there exists a postselection oracle algorithm \( A : U(d) \to L(H \otimes K) \), which \( \epsilon \)-approximates the \( \frac{1}{d} \)-th power task. Use \( A(U) \) \( d \) times as a subroutine in the \( c-U^d \) algorithm of Dong et al. [DNSM19] (see Figure 3 and eq. (4)). In other words, replace its \( \text{id} \) queries by \( A \) queries. The result is illustrated in Figure 4. The intuition is that \( A(U) \) is close to \( U^\frac{1}{d} \otimes G_U \) for some garbage operator \( G_U \) on the ancilla, and therefore the postselection oracle algorithm of Figure 4 should be appropriately close to implementing
\[
\left( |0\rangle_0 \otimes 1 + \frac{1}{\det(U^\frac{1}{d})} |1\rangle_1 \otimes U \right) \otimes (G_U |0\rangle)^\otimes d \otimes \det(U^\frac{1}{d}) |0\rangle |0\rangle^\otimes d,
\]
which corresponds to the task operator of the task \( c-U \). Indeed, by Theorems 2.1 and 2.2 of [Gav20], the algorithm of Figure 4 with its \( d \) subroutines \( de' \)-approximates \( c-U \), where \( e' = O(\sqrt{\epsilon}) \). Since by Theorem 2 we must have \( de' > \frac{1}{2} \), we get \( \epsilon = \Omega(d^{-2}) \). \( \square \)
where $U$.

![Figure 4: The c-U$^d$ algorithm of [DNSM19] with the query sequence $s = (id)^d$ replaced by $s' = A^d$, where $A(U)$ approximately implements $U^\pi \otimes G_U$.]

### 5.2 An efficient average-case c-U algorithm?

If we substitute the average-case $1/k$-th power algorithm of [SMM09] for $A(U)$ in Figure 4 we obtain an average-case c-U algorithm. The [SMM09] algorithm approximately implements $U^\pi \otimes U^{-\pi} \otimes 1$ for most $U \in U(d)$ (see Section 2.5) and its oracle complexity is constant in $d$. However, it is used in Figure 4 $d$ times so the resulting algorithm is exponential in $n := \lceil \log(d) \rceil$, the number of qubits $U \in U(d)$ acts on.

It remains open whether an efficient average-case algorithm for c-U exists. A naive idea is to go through the $1/k$-th power instead of the $1/d$-th power for some $k = poly(n)$ and to replace the c-U$^d$ algorithm of Dong et al. by an algorithm that makes only $k$ calls to the $1/k$-th power subroutine. However, Theorem 2 implies that a worst-case $c-U^k$ algorithm for such $k$ is impossible, so we have to find a different candidate for replacing Dong et al.’s algorithm.

Interestingly, such a candidate may exist and may even still be worst-case. Our impossibility of $c-U^k = (U \mapsto control_\phi(U^k), \{id, inv\})$ for $k = poly(n)$, $k \neq 1$ does not imply the impossibility of, for example, the task $(control_\phi, \{root_k\})$, where $\text{root}_k(U) = U^{\frac{\pi}{k}}$ picks some $k$-th root of $U$. Intuitively, an algorithm could achieve the new task by exploiting the additional knowledge of being given $k$-th roots of $U \in U(d)$. In any case, as we argue next, Theorem 2 cannot be extended to exclude such an algorithm. Note that any algorithm $A$ whose query sequence is $s = (\text{root}_k)^k$ can be decomposed as $A = A' \circ \text{root}_k$ where $A'$ is an algorithm with the query sequence $(id)^k$.

We could try to derive an impossibility for either $A$ or $A'$ by reducing to a function $f_A$ or $f_{A'}$ respectively, which meets the conditions of Lemma 1. However, an $f_A$ derived from $A$ is not necessarily continuous since $\text{root}_k$ is not continuous. On the other hand, an $f_{A'}$ derived from $A'$ has $\text{Domain}(f_{A'}) = \text{Domain}(A') = \text{Im}(\text{root}_k) \subseteq U(d)$ because any $\text{root}_k : U(d) \to U(d)$ picking some $k$-th root of $U$ cannot be onto. Therefore this simple example of a route towards an efficient average-case c-U remains open, as do many others.

### 6 Discussion: Limitation of our model

One may ask whether our postselection oracle algorithm (defined in Section 2.1) is the most general model one can define, for an algorithm making use of an unknown unitary. The answer, surprisingly, is quite subtle. It turns out that one can conceive of a more general model: one in which the postselection measurements have more than one success possibility. We call those non-binary postselection measurements. We will see that for many tasks, this generalization does not make any difference. However, in relation to c-U, it does!

We represent a postselection oracle algorithm by a function $A$ from the set of unitary oracles
$U(d)$ to the set of operators the algorithm implements conditioned on getting 'success'. The function is deterministic and continuous (Property 1). This picture is appropriate for computations that include the following two measurement scenarios

I. measure and accept only a specific 'success' outcome (project$^2$),
II. measure and forget the outcome (trace out).

The function $A$ includes a multiplication by a projector whenever a type I measurement occurs and ignores all type II measurements. However, a third measurement scenario exists - the non-binary postselection measurement scenario:

III. measure and accept any outcome from the 'success set', remembering$^3$ which of them was actually obtained.

This scenario reduces to scenario I if the success set has one element. Otherwise, a computation with a measurement of type III is outside our deterministic model. We argue next that the process-tomography strategy to approximate $c-U$ is an example. Afterwards we explain why this distinction does not make any difference for some other tasks, such as the complex conjugation, transpose and the inverse tasks.

6.1 $c-U$ via process tomography uses a type III measurement

The process tomography strategy to approximate $c-U$, like any quantum-classical algorithm, can in principle be made all unitary with a measurement at the end. The measurement is without loss of generality of the form $\{|x\rangle \langle x|\}_{x \in \text{dim}(K_m)}$, where $K_m$ is the measured register and the integers $x$ label vectors in a fixed orthonormal basis of $K_m$. Assume for simplicity that in this $c-U$ algorithm all outcomes $x$ are in the 'success set'. Then, before the measurement the algorithm must be close$^4$ to implementing the following operator in $L(H_t \otimes K_{\text{rest}} \otimes K_m)$:

$$\sum_{x \in \text{dim}(K_m)} (|0\rangle \langle 0| \otimes 1 + e^{i\phi_x(U)} |1\rangle \langle 1| \otimes U) \otimes G_{U,x} \otimes |x\rangle |0\rangle,$$

(10)

provided that the ancilla Hilbert spaces $K_{\text{rest}} \otimes K_m$ were initialised to the all-zero state at the onset of the algorithm. Then $||G_{U,x} |0\rangle||^2$ is the probability of outcome $x$. So far the algorithm, seen as a function from oracles $U(d)$ to the implemented operators, is continuous (Property 1).

Our claim that the process tomography strategy must contain a type III measurement is equivalent to the following

Claim 2. The process tomography strategy for $c-U$ corresponds to an operator (10) with at least two terms in the sum.

Proof and intuition. Assume towards contradiction that there is only one term ($x = 1$) in the sum. Then the measurement can be omitted. We have reduced the $c-U$ algorithm to a continuous (postselection oracle) algorithm!

By Theorem 2 this is impossible.

We can choose not to invoke Theorem 2, and instead try to find how the above contradicts the process tomography strategy in particular. This will, in turn, give us some intuition for Theorem 2. Let's go back to the process tomography strategy before we reduced it to a fully quantum algorithm without intermediate measurements. There the measurements of the process-tomography step yield the matrix elements of some representative of $U$ disregarding the global phase. The assumption of the sum (10) having only one, $x = 1$ term translates to this 'picking of representative' being close to a deterministic function of $U$, namely to the function $\text{rep}(U) = e^{i\phi(U)} U$. At the same time, the insensitivity to the global phase imposes

---

$^2$after possibly adding some ancillae and appropriate rotations, to account for a general POVM
$^3$otherwise this would just be a combination of scenarios I and II.
$^4$Equation (9) justifies the use of operators here instead of superoperators.
that \( \text{rep}(U) = \text{rep}(e^{i\alpha}U) \) for all \( \alpha \in [0, 2\pi) \). An example of such a \( \text{rep} \) function is the one with its \( e^{i\phi(U)} \) tuned to make the first nonzero matrix element encountered, when going over them in some fixed order, real positive. This example of a \( U \mapsto e^{i\phi(U)} \) map (and a \( \text{rep} \) map) is discontinuous. Is there a continuous one? By Lemma 1 there isn’t! \( \text{rep}(U) = \text{rep}(e^{i\alpha}U) \) imposes that \( U \mapsto e^{i\phi(U)} \) be \((-1)\)-homogeneous. Therefore, a continuous algorithm approximates a discontinuous function with an arbitrarily small approximation error \(^5\). This is impossible! The inconsistency can only be removed by letting \( ||G_{U,1}|0\rangle||^2 \) go to zero for \( U \)'s at the discontinuity. However, since the algorithm is worst-case, a new term’s \( ||G_{U,x}|0\rangle||^2 \) must increase, hence the claim.

### 6.2 For some tasks our model is general enough

The crucial difference between the \( c-U \) task and the complex conjugation, transpose and inverse tasks is that \( c-U = \{\text{control}, \{\text{id, inv}\}\} \) has some freedom in its task function. In the sum (10) each term has a different phase function, i.e. \( \phi_x \neq \phi_{x'} \) for \( x \neq x' \), while each term on its own 'succeeds'. To see this, note that if \( \phi_x = \phi_{x'} \), then without loss of generality the terms \( x \) and \( x' \) can be grouped into one term. Similarly, if any type III measurement is present in, for example, an inverse algorithm, then, since the inverse is a unique function, we are left with a single ('success') term

\[
\sum_{x} U^\dagger \otimes G_{U,x} \otimes |x\rangle\langle 0| = U^\dagger \otimes \sum_{x} G_{U,x} \otimes |x\rangle\langle 0| = U^\dagger \otimes G'_U
\]

and the measurement \( \{|x\rangle\langle x|\}_x \) can be skipped. Thus the algorithms of [MSM19, QDS+19a, QDS+19b] are postselection oracle algorithms, as any other algorithms achieving those tasks would be. Interestingly, the algorithm of [DNSM19] is also within our model and the [SMM09] algorithm differs only by being average-case, while the corresponding tasks, the \( c-U^d \) and the \( q\)-th power task, do have freedom in their task functions. Therefore, solutions outside our model may exist as well. We leave this as an open question, together with an efficient, type-III algorithm for \( c-U \).

**Acknowledgements**

The authors would like to thank Dorit Aharonov and Amitay Kamber for useful discussions.

---

\(^5\)In process tomography any approximation error is attainable at the cost of increasing the complexity of the procedure.
References

[Aar05] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 461(2063):3473–3482, 2005.

[AFCB14] Mateus Araújo, Adrien Feix, Fabio Costa, and Časlav Brukner. Quantum circuits cannot control unknown operations. New Journal of Physics, 16(9):093026, 2014.

[AK07] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In Twenty-Second Annual IEEE Conference on Computational Complexity (CCC’07), pages 115–128. IEEE, 2007.

[AKN98] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. Quantum circuits with mixed states. In Proceedings of the thirtieth annual ACM symposium on Theory of computing, pages 20–30, 1998.

[BBC+95] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. Physical review A, 52(5):3457, 1995.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. Quantum lower bounds by polynomials. Journal of the ACM (JACM), 48(4):778–797, 2001.

[BDW02] Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. Theoretical Computer Science, 288(1):21–43, 2002.

[Bor33] Karol Borsuk. Drei sätze über die n-dimensional euklidische sphäre. Fundamenta Mathematicae, 20(1):177–190, 1933.

[BS17] Fernando GSL Brandão and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), pages 415–426. IEEE, 2017.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 454(1969):339–354, 1998.

[CN97] Isaac L Chuang and Michael A Nielsen. Prescription for experimental determination of the dynamics of a quantum black box. Journal of Modern Optics, 44(11-12):2455–2467, 1997.

[DNSM19] Qingxiuxiong Dong, Shojun Nakayama, Akihito Soeda, and Mio Murao. Controlled quantum operations and combs, and their applications to universal controlization of divisible unitary operations. arXiv preprint arXiv:1911.01645, 2019.

[FDDB14] Nicolai Friis, Vedran Dunjko, Wolfgang Dür, and Hans J Briegel. Implementing quantum control for unknown subroutines. Physical Review A, 89(3):030303, 2014.

[FMKB15] Nicolai Friis, Alexey A Melnikov, Gerhard Kirchmair, and Hans J Briegel. Coherent controlization using superconducting qubits. Scientific reports, 5(1):1–11, 2015.

[Gav20] Zuzana Gavorová. Notes on distinguishability of postselected computations, 2020.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix
arithmetics. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, pages 193–204, 2019.

[Hah50] Erwin L Hahn. Spin echoes. Physical review, 80(4):580, 1950.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical review letters, 103(15):150502, 2009.

[Kit95] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. arXiv preprint quant-ph/9511026, 1995.

[MS17] Dusa McDuff and Dietmar Salamon. Introduction to symplectic topology. Oxford University Press, 2017.

[MSM19] Jisho Miyazaki, Akihito Soeda, and Mio Murao. Complex conjugation supermap of unitary quantum maps and its universal implementation protocol. Physical Review Research, 1(1):013007, 2019.

[Mun00] James R. Munkres. Topology. Prentice Hall, Inc., Upper Saddle River, NJ, 2000. Second edition of [ MR0464128].

[PCZ97] JF Poyatos, J Ignacio Cirac, and Peter Zoller. Complete characterization of a quantum process: the two-bit quantum gate. Physical Review Letters, 78(2):390, 1997.

[QDS⁺19a] Marco Túlio Quintino, Qingxiuxiong Dong, Atsushi Shimbo, Akihito Soeda, and Mio Murao. Probabilistic exact universal quantum circuits for transforming unitary operations. Physical Review A, 100(6):062339, 2019.

[QDS⁺19b] Marco Túlio Quintino, Qingxiuxiong Dong, Atsushi Shimbo, Akihito Soeda, and Mio Murao. Reversing unknown quantum transformations: Universal quantum circuit for inverting general unitary operations. Physical Review Letters, 123(21):210502, 2019.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science, pages 124–134. Ieee, 1994.

[SMM09] Lana Sheridan, Dmitri Maslov, and Michele Mosca. Approximating fractional time quantum evolution. Journal of Physics A: Mathematical and Theoretical, 42(18):185302, 2009.

[TGMV13] Jayne Thompson, Mile Gu, Kavan Modi, and Vlatko Vedral. Quantum computing with black-box subroutines. arXiv preprint arXiv:1310.2927, 2013.

[TOV⁺11] Kristan Temme, Tobias J Osborne, Karl G Vollbrecht, David Poulin, and Frank Verstraete. Quantum metropolis sampling. Nature, 471(7336):87–90, 2011.

[VAGGdW20] Joran Van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. Quantum, 4:230, 2020.

[WBAG11] James D Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure hamiltonians using quantum computers. Molecular Physics, 109(5):735–750, 2011.

[ZRK⁺11] Xiao-Qi Zhou, Timothy C Ralph, Pruet Kalasuwan, Mian Zhang, Alberto Peruzzo, Benjamin P Lanyon, and Jeremy L O’Brien. Adding control to arbitrary unknown quantum operations. Nature communications, 2(1):1–8, 2011.
Appendices

A Special-case impossibility via the Borsuk-Ulam theorem

Here we prove the impossibility of \( c-U \) for even-dimensional oracles from the notorious result of algebraic topology: the Borsuk-Ulam theorem \cite{Bor33}. This shows that our Lemma 1 can be seen as a variant of the Borsuk-Ulam theorem.

**Borsuk-Ulam theorem.** If \( f : S^n \to \mathbb{R}^n \) is a continuous function, then there exists \( x \in S^n \) such that \( f(x) = f(-x) \).

**Proof of Corollary 1 (d even).** Assume \( A : U(d) \to L(\mathcal{H} \otimes \mathcal{K}) \) is a postselection oracle algorithm exactly achieving \( c-U \). In fact, any postselection oracle algorithm making queries from \( \Sigma = \{id, inv\} \)
- is continuous (see Property 1),
- is an odd (even) function if the total number of queries \( |s| \) is odd (even).

Define \( h(U) := \langle 0_c | (U \otimes 1_K) \left( (|1_c \rangle A(U) \langle 1_c |)^\dagger (\langle 0_c | A(U) | 0_c \rangle) | 0 \rangle \right| \),
where \( |0_c \rangle \) and \( |1_c \rangle \) are the computational basis states of the first qubit and \( |0 \rangle \) the all-zero state on what is left (i.e. if \( |0 \rangle \) is the all-zero state over all registers, then \( |0_c \rangle \otimes |0 \rangle = |0 \rangle \)). From the listed properties of \( A \) it follows that \( h \) is a continuous odd function, i.e. \( h(-U) = -h(U) \).

Next we use the fact that \( A \) exactly achieves the \( c-U \) task, i.e. there is a real function \( \phi \) and a \( U \)-dependent ancilla operator \( G_U \in L(\mathcal{K}) \) such that
\[
A(U) \left( \mathbf{1}_{\mathcal{H}} \otimes |0 \rangle \langle 0 | \right) = \left( |0 \rangle |0_c \rangle \otimes \mathbf{1} + e^{i\phi(U)} |1 \rangle |1_c \rangle \otimes U \right) \otimes G_U.
\]
Substituting this into the definition of \( h \), we get
\[
h(U) = e^{-i\phi(U)} |0_c \rangle G_U |0 \rangle |0 \rangle = e^{-i\phi(U)} |0 \rangle A(U)^\dagger A(U) |0 \rangle,
\]
i.e., the amplitude of \( h(U) \) is the postselection probability when the input is \( |0 \rangle \in \mathcal{H} \otimes \mathcal{K} \) and therefore is nonzero for all \( U \in U(d) \). In other words, \( \text{Im}(h) \subseteq \mathbb{C} \setminus \{0\} \). Now, since \( d \) is even we can define the following continuous function \( g : S^3 \to U(d) \) such that given a vector \( x = (x_1, x_2, x_3, x_4)^T \) on the 3-sphere
\[
g(x) = \begin{pmatrix}
x_1 + ix_2 \\
\vdots \\
x_3 + ix_4
\end{pmatrix}
\begin{pmatrix}
x_1 + ix_2 \\
x_3 + ix_4
\end{pmatrix}^T
\begin{pmatrix}
x_1 + ix_2 \\
x_3 + ix_4
\end{pmatrix}
\]
where the dots denote that the elements are repeated across the diagonal and the antidiagonal, and there are zeroes everywhere else. Note that \( g \) is odd, \( g(-x) = -g(x) \). By Borsuk-Ulam theorem, since \( h \circ g : S^3 \to \mathbb{C} \subseteq \mathbb{R}^3 \) is continuous, there exists \( x \in S^3 \) such that \( h \circ g(x) = h \circ g(-x) = -h \circ g(x) \), where the last equality holds because \( h \circ g \) is odd. Therefore there exist \( x \in S^3 \) such that \( h \circ g(x) = 0 \). This is in contradiction with \( \text{Im}(h) \subseteq \mathbb{C} \setminus \{0\} \).

In this appendix we decided to discuss the impossibility of \( c-U \) i.e. the \( m = 1 \) case of Theorem 1, but note that only minor modifications of the above proof are needed to generalise it to \( c-U^m \) for \( m \) odd.
B \(c-U^d\) from the simplest inversion algorithm

In this section we present an algorithm for \(c-U^d\) built from the simplest of the family of inversion algorithms due to [QDS+_19b]. This may serve as an alternative proof of the existence direction of Theorems 1 and 2, as well as an illustration of the construction used in the proof of Corollary 3. First, we review the results leading to the simplest inversion algorithm, then we present the \(c-U^d\) construction.

B.1 Review of the complex conjugation algorithm

Here we review the complex conjugation algorithm of Miyazaki et al. [MSM19] with an alternative proof of its correctness.

**Construction 2 (due to [MSM19]).** There exists a unitary \(V_{\text{conj}}\) such that the dashed box in Figure 5 represents a clean unitary oracle algorithm exactly achieving the complex conjugation task \((U \mapsto U^*, \{id\})\).

In particular, the dashed box corresponds to \(A_* : U(d) \mapsto L(\mathcal{H}_t \otimes \mathcal{K}_s)\) with a single qudit task register \(\mathcal{H}_t\) and a \((d-2)\)-qudit ancilla register \(\mathcal{K}_s\), such that

\[
A_* (U) \left( 1 \otimes |0\rangle \langle 0| ^{\otimes d-2} \right) = U^* \otimes \det(U) |0\rangle \langle 0| ^{\otimes d-2} \tag{11}
\]

**Proof.** Cramer’s rule for matrix inversion reads \(U^{-1} = \det(U)^{-1} C(U)^T\), where \(C(U)\) is the cofactor matrix of \(U\). We define it presently, but first note that from Cramer’s rule \(C(U) = \det(U) U^*\), so \(C(U)\) is unitary and it is enough to show that the algorithm implements \(C(U)\). Cofactor matrix of \(U\) is the matrix of minors of \(U\), \(C(U)_{i,j} = (-1)^{i+j} \det(U_{\cdot \cdot})\), or in the quantum notation (with the first basis vector labeled by 1):

\[
C(U) = \sum_{i,j=1}^{d} (-1)^{i+j} \det(U_{\cdot \cdot}) |i\rangle \langle j| ,
\]

where \(\det(U_{\cdot \cdot})\) is the \((i,j)\)-th minor, defined as the determinant of the matrix \(U\) with the \(i\)-th
row and \( j \)-th column deleted, which can be expressed as

\[
\det(U_{\rightarrow j}) = \frac{(-1)^{i+j}}{(d-1)!} \sum_{\pi \in S_d} \operatorname{sgn}(\pi) \prod_{k=2}^{d} \langle \tau(k) | U | \pi(k) \rangle.
\] (12)

See Appendix C for the proof. Substituting this into the cofactor matrix definition we get

\[
C(U) = E^\dagger U^\otimes (d-1) E
\]

where \(|v_j\rangle := E |j\rangle\). Note that \(\{|v_j\rangle\}_{j \in [d]}\) is an orthonormal basis of the \((d-1)\)-qudit Hilbert space. To avoid conflation next, let’s go back the the traditional labelling of basis vectors, which starts with \(j = 0\). Choose a unitary \(V_{\text{conj}}\) such that \(V_{\text{conj}}(|j⟩ \otimes |0⟩^\otimes (d-2)) = |v_j⟩ = E |j⟩\), which implies \(V_{\text{conj}}(1 \otimes |0⟩^\otimes (d-2)) = E\). Because of the orthonormality of \(\{|v_j\rangle\}_{j \in \{0,1,...,d-1\}}\) such a unitary exists. The dashed box in Figure 5 then implements \(A_o(U) := V_{\text{conj}} U^\otimes (d-1) V_{\text{conj}}\), which on input \(|ξ⟩ \otimes |0⟩^\otimes (d-2)\) for any \(|ξ⟩ \in H_t\) outputs the all-zero state on the ancilla with probability

\[
p = \left|\left|E^\dagger U^\otimes (d-1) |ξ⟩\right|\right|^2 = ||C(U) |ξ⟩||^2 = 1.
\]

Therefore projecting the ancilla output onto the all-zero state has no effect and

\[
A_o(U) \left(1 \otimes |0⟩^\otimes (d-2)\right) = \left(1 \otimes |0⟩^\otimes (d-2)\right) V_{\text{conj}}^\dagger U^\otimes (d-1) V_{\text{conj}} (1 \otimes |0⟩^\otimes (d-2))
\]

\[
= C(U) \otimes |0⟩^\otimes (d-2) = U^* \otimes \det(U) |0⟩^\otimes (d-2),
\]

confirming eq. (11). \(\square\)

### B.2 Review of the simplest transpose and inverse algorithms

Quintino et al. [QDS+19b, QDS+19a] presented a family of postselection oracle algorithms exactly achieving the transpose task and, when composed with the complex conjugation algorithm of [MSM19], the inverse task. Here we review the simplest transpose algorithm which, making only one query, succeeds with the lowest probability.

**Construction 3.** The algorithm of Figure 5, with the dashed box replaced by one \(\{\text{id}\}\) query to \(U \in U(d)\) is (up to the initialisation of the ancilla) a postselection oracle algorithm exactly achieving the transpose task \((U \mapsto U^T, \{\text{id}\})\).

This construction together with Construction 2 give

**Construction 4.** The algorithm of Figure 5 is (up to the initialisation of the ancilla) a postselection oracle algorithm exactly achieving the inverse task \((U \mapsto U^\dagger, \{\text{id}\})\).

**Proof of Construction 3.** Note that if the dashed box is replaced by the identity, Figure 5 corresponds to the generalised teleportation, which succeeds with probability \(\frac{1}{d^2}\). In other words
\[ ((\psi^+ \otimes \mathbb{1}) (\mathbb{1} \otimes |\psi^\rangle)) = \frac{1}{\sqrt{d}} \mathbb{1}, \] which is easy to check by substituting for the generalised Bell state \(|\psi^\rangle\) = \( \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} |ii\rangle \). Next, for any complex \( d \times d \) matrix \( M \),

\[
(M \otimes \mathbb{1}) \sum_{i=0}^{d-1} |ii\rangle = \mathbb{1} \otimes M^T \sum_{i=0}^{d-1} |ii\rangle.
\]

which can be verified by writing \( M \) in terms of its matrix elements. Combining the two observations we get that the algorithm of Figure 5 with the dashed box replaced by \( U \) implements

\[
((\psi^+ \otimes \mathbb{1}) (\mathbb{1} \otimes U \otimes \mathbb{1}) (\mathbb{1} \otimes |\psi^\rangle)) = ((\psi^+ \otimes \mathbb{1}) (\mathbb{1} \otimes U^T) (\mathbb{1} \otimes |\psi^\rangle)) = U^T ((\psi^+ \otimes \mathbb{1}) (\mathbb{1} \otimes |\psi^\rangle)) = \frac{1}{d} U^T
\]
as claimed.

B.3 The c-\( U^d \) construction

Theoretically, any clean \( k \)-query exact inversion algorithm could be used to build an algorithm exactly achieving c-\( U^{k+1} \). The general construction is described in the proof of Corollary 3.1. As an example, we compose the clean \( d - 1 \)-query inversion algorithm of Figure 5 coherently into the larger algorithm of Figure 6. Note that the algorithm of Figure 6 is reminiscent of the Hahn spin echo trick [Hah50], because when the control is \(|0\rangle\) it "corrects" the first query to \( U \).

Construction 5. Figure 6 represents a postselection oracle algorithm exactly achieving the task c-\( U^d \) = \((U \mapsto \text{control}_\phi(U^d), \{id, inv\})\).

Proof. Observe that the dashed box inside fig. 6 implements \(|0\rangle\langle 0| \otimes \mathbb{1} \otimes A_s(U) + |1\rangle\langle 1| \otimes U^\otimes d-1 \otimes \mathbb{1}\), where \( A_s \) satisfies eq. (11). The initialisation of the ancilla and the postselection measurements at the end transpose a part of the dashed box resulting in the operator \( \frac{1}{d} (|0\rangle\langle 0| \otimes \det(U)U^T + |1\rangle\langle 1| \otimes U^\otimes d-1) \). Together with the initial query to \( U \) the algorithm exactly achieves c-\( U^d \).

C Symmetric equations for the determinant and the minors

In the following we first prove the symmetric formula for the determinant of eq. (5), then we use it to prove Lemma 4, the symmetric formula for the minors of eq. (12).
Lemma 3. The determinant of $M \in \text{Mat}_{n \times n} (\mathbb{C})$ is given by the formula:

$$
\det (M) = \frac{1}{n!} \sum_{\pi, \tau \in S_n} \text{sgn} (\tau) \text{sgn} (\pi) \prod_{i=1}^{n} M_{\tau(i), \pi(i)}.
$$

Proof. By the usual Leibniz formula, we have $\det (M) = \sum_{\pi \in S_n} \text{sgn} (\pi) \prod_{i=1}^{n} M_{\pi(i)}$. For each $\tau \in S_n$, set $M^\tau$ to be the matrix obtained from $M$ by permuting the rows of $M$ according to $\tau$, i.e. $M^\tau_{ij} = M_{\tau(i), \tau(j)}$. On the one hand, by the properties of the determinant function we have $\det (M^\tau) = \text{sgn} (\tau) \det (M)$, and on the other hand by the Leibniz formula:

$$
\det (M^\tau) = \sum_{\pi \in S_n} \text{sgn} (\pi) \prod_{i=1}^{n} M^\tau_{\pi(i)} = \sum_{\pi \in S_n} \text{sgn} (\pi) \prod_{i=1}^{n} M_{\tau(i), \pi(i)}.
$$

Combining the two expressions for $\det (M^\tau)$ and multiplying by $\text{sgn} (\tau)$ gives

$$
\det (M) = \sum_{\pi \in S_n} \text{sgn} (\pi) \text{sgn} (\tau) \prod_{i=1}^{n} M_{\tau(i), \pi(i)}.
$$

Summing this equality over all $\tau \in S_n$ and dividing by $n!$ gives the desired result. \hfill \Box

Lemma 4. The $(i, j)$ minor a matrix $M \in \text{Mat}_{n \times n} (\mathbb{C})$ is given by

$$
\det (M_{i,j}) = \frac{(-1)^{i+j}}{(n-1)!} \sum_{\pi, \tau \in S_n} \text{sgn} (\tau) \text{sgn} (\pi) \prod_{k=2}^{n} M_{\tau(k), \pi(k)}.
$$

Proof. We think of $S_{n-1}$ as the subset of $S_n$ of permutations fixing $n$, i.e. $S_{n-1} = \{ \alpha \in S_n \mid \alpha (n) = n \}$.

For each $m \in \{ i, j \}$ set $X_m = \{ \alpha \in S_n \mid \alpha (1) = m \}$, and let $T_m : X_m \to S_{n-1}$ be the mapping $T_m (\alpha) = (n \, n-1 \, ... \, m) \circ \alpha \circ (1 \, 2 \, 3 \, ... \, n-1 \, n)$. Note that $T_m (\alpha)$ lies in $S_{n-1}$ since it fixes $n$, that $T_m$ is a bijection (since $T_m \alpha$ it is obtained from $\alpha$ by composing on both sides with fixed permutations), and finally that by the multiplicativity of $\text{sgn}$ we have

$$
\text{sgn} (T_m (\alpha)) = (-1)^{n-m} \cdot \text{sgn} (\alpha) \cdot (-1)^{n-1} = (-1)^{1-m} \cdot \text{sgn} (\alpha).
$$

One can now verify that for all $k \in \{1, 2, ..., n-1\}$ and for every $\tau \in X_i$ and $\pi \in X_j$ we have:

$$(M_{i,j})_{T_i \tau(k), T_j \pi(k)} = M_{\tau(k+1), \pi(k+1)}.
$$

Indeed, if $\tau (k+1) < i$ then deleting the $i$-th row from $M$ doesn’t change the index of the $\tau (k+1)$-th row, and correspondingly $(n \, n-1 \, ... \, i)$ fixes $\tau (k+1)$ and so $(T_i \tau) (k) = \tau (k+1)$. If $\tau (k+1) > i$ then then deleting the $i$-th row from $M$ decreases the index of the $\tau (k+1)$-th row by one, and correspondingly $(n \, n-1 \, ... \, i)$ does the same thing to $\tau (k+1)$. A similar argument for $\pi$ shows the validity of the last equality. Now, by Lemma 3:

$$
\det (M_{i,j}) = \frac{1}{(n-1)!} \sum_{\pi', \tau' \in S_{n-1}} \text{sgn} (\tau') \text{sgn} (\pi') \prod_{k=1}^{n-1} (M_{i,j})_{\tau'(k), \pi'(k)} =
$$

$$
= \frac{1}{(n-1)!} \sum_{\pi \in X_i} \sum_{\tau \in X_j} \text{sgn} (T_i \tau) \text{sgn} (T_j \pi) \prod_{k=1}^{n-1} (M_{i,j})_{T_i \tau(k), T_j \pi(k)} =
$$

$$
= \frac{1}{(n-1)!} \sum_{\pi \in X_i} \sum_{\tau \in X_j} (-1)^{1-i} \text{sgn} (\tau) (-1)^{1-j} \text{sgn} (\pi) \prod_{k=1}^{n-1} M_{\tau(k+1), \pi(k+1)} =
$$

$$
= \frac{(-1)^{i+j}}{(n-1)!} \sum_{\pi \in X_i} \sum_{\tau \in X_j} \text{sgn} (\tau) \text{sgn} (\pi) \prod_{k=2}^{n} M_{\tau(k), \pi(k)}.
$$

\hfill \Box

25