Feynman Meets Turing: Computability Aspects of Quantum Compiling Revisited

Yannik N. Böck, *Graduate Student Member, IEEE*, Holger Boche, Zoe Garcia del Toro, *Fellow, IEEE*, and Frank H.P. Fitzek, *Fellow, IEEE*

***Abstract*—We consider a formalism of *quantum compiler functions* – functions that map unitary matrices to corresponding gate-circuit approximations – and prove the infeasibility of digitally computing such functions. Since the gate-circuit model of quantum computing emerged, much research has been conducted to find algorithmic solutions to the *quantum compiler problem*. The renowned *Solovay-Kitaev theorem* proves the existence of quantum compiler functions that provide low-complexity gate-circuit approximations to arbitrary unitary matrices, which is indispensable for the practical feasibility of gate-based quantum computing. However, the mere existence of such functions does not imply their *realizability* by means of an *algorithm* – a constructive procedure executed by a *Turing machine*. In fact, no algorithm for computing any quantum compiler function is known today. The present article demonstrates that no such algorithm can exist. We prove that no quantum compiler function can satisfy *Banach-Mazur computability*, which is a formalization of algorithmic feasibility with (mathematically) weak requirements. In consequence, there definitely does not exist a Turing machine that computes any quantum compiler function in the above sense, nor can there exist a *constructive proof* of the existence of any such function. Furthermore, we discuss proposed methods of quantum compiling and analyze them in the context of our results.**

***Index Terms*—Quantum computing, quantum compiling, Solovay-Kitaev theorem, Turing machine, computable analysis.**

Received 11 April 2025; revised 1 September 2025; accepted 19 October 2025. Date of publication 28 October 2025; date of current version 15 January 2026. The work of Yannik N. Böck and Holger Boche was supported in part by the German Federal Ministry of Education and Research (BMBF) within the National Initiative on 6G Communication Systems through the Research Hub 6G-life under Grant 16KISK002. The work of Yannik N. Böck was supported in part by the BMBF Quantum Programm QD-CamNetz under Grant 16KISQ077, in part by QuaPhySI under Grant 16KIS1598K, and in part by QUIET under Grant 16KISQ093. The work of Zoe Garcia del Toro was supported in part by the German Research Foundation (DFG) within the Gottfried Wilhelm Leibniz Prize Programme under Grant BO 1734/20-1 and DFG Projects BO 1734/35-1 and STA 864/9-1. The work of Frank H.P. Fitzek was supported in part by the German Research Foundation (DFG, Deutsche Forschungs- gemeinschaft) as part of Germany's Excellence Strategy – EXC2050/1 – Project 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden. This work was supported in part by the Federal Ministry of Education and Research of Germany in the program "Souverän. Digital. Vernetzt." – Joint Project 6G-life, project identification number 16KISK001K; Project QUARKS, Project 16KIS1998K; Project Q-TREX, Project 16KISR027; and Project QUIET, Project 16KISQ092. An earlier version of this paper was presented at the IEEE International Conference on Communications (ICC) 2024 [DOI: 10.1109/ICC51166.2024.10622486]. Recommended for acceptance by M. Palesi. *(Corresponding author: Yannik N. Böck.)*

Yannik N. Böck is with the Chair of Theoretical Information Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: yannik.boeck@tum.de).

Holger Boche is with Chair of Theoretical Information Technology, Technical University of Munich, 80333 Munich, Germany, also with the Munich Center for Quantum Science and Technology (MCQST), D-80799 Munich, Germany, Munich Quantum Valley (MQV), D-80807 Munich, Germany, and also with BMBF Research HUB 6G-life, D-80333 Munich, Germany (e-mail: boche@tum.de).

Zoe Garcia del Toro is with LIP6, CNRS, Sorbonne Université, F-75005 Paris, France (e-mail: zoe.garcia@lip6.fr).

Frank H.P. Fitzek is with Deutsche Telekom Chair of Communication Networks, Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) D-01062 Dresden, Germany, and also with BMBF Research HUB 6G-life, D-80333 Munich, Germany, Technical University of Dresden, D-01062 Dresden, Germany (e-mail: frank.fitzek@tu-dresden.de).

Digital Object Identifier 10.1109/TC.2025.3626469

I. Introduction

QUANTUM computing is considered one of the potentially most disruptive technologies of the 21st century. Theoretical findings suggest that quantum computers can solve specific computational problems, such as the simulation of protein behavior or the factorization of large integers, in substantially less time than classical digital computers [2]. The scientific community expects quantum computers to significantly increase global computing power for various domains and applications, including computational chemistry and physics, logistics, cryptography, and economics. Nevertheless, building quantum computers that can perform the required computations on a practically relevant scale proves challenging. Aside from many unresolved obstacles in manufacturing scalable and robust quantum hardware, the theory of quantum algorithms has grown much more slowly than expected. *Peter W. Shor*, inventor of the renowned *Shor's algorithm* [3], [4] for prime factorization, notes that the actual progress in the development of quantum algorithms has been diminishing as compared to the expectations his seminal publications [3], [4] raised within the community of quantum engineering [5], [6]. To the best of the authors' knowledge, only a minority of all proposed quantum algorithms - including [3], [4] - provide error bounds and convergence guarantees that meet the standards of mathematical computing theory.

For $t \in \mathbb{R} \cap [0,1]$, a noiseless quantum-physical system consists of a Hilbert space $\mathcal{H}$ – throughout this article, we will consider Hilbert spaces of *finite dimension* –, a time-dependent state vector $|\phi_t\rangle \in \mathcal{H}$, and a time-dependent skew-Hermitian operator $\boldsymbol{H}_t : \mathcal{H} \to \mathcal{H}$, which we call the system's *Hamiltonian*.

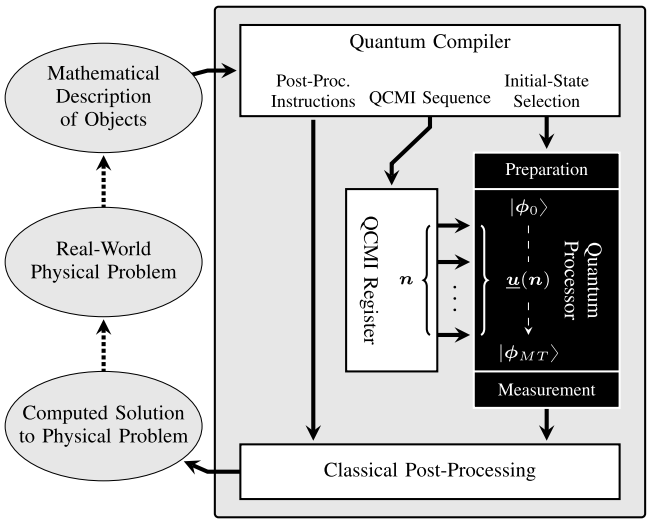

Figure 1. *Schematic representation of quantum computing in the gate-circuit model*. Based on a target unitary operator $\boldsymbol{U}$ (*quantum algorithm*) and a prescribed accuracy $L \in \mathbb{N}$, digital hardware computes a symbolic sequence $\boldsymbol{n} := (n_1, \ldots, n_M)$ of *quantum-circuit machine instructions* (QCMI), and successively applies their physical counterparts. The resulting unitary transformation $\underline{\boldsymbol{u}}(\boldsymbol{n}) = \underline{\boldsymbol{u}}(n_m) \cdots \underline{\boldsymbol{u}}(n_1)$ evolves any initial state $|\phi_0\rangle$ into a new state $|\phi_{MT}\rangle$ that satisfies $\|\boldsymbol{U}|\phi_0\rangle - |\phi_{MT}\rangle\| < 1/2^L$.

The state vector then satisfies *Schrödinger's equation* (in integral form),

$$|\phi_t\rangle = \int_0^t \boldsymbol{H}_\tau |\phi_\tau\rangle \mathrm{d}\tau. \tag{1}$$

If the system's Hamiltonian is time *independent* – i.e., if we have $\boldsymbol{H}_\tau = \boldsymbol{H}_\kappa =: \boldsymbol{H}$ for all $\tau, \kappa \in \mathbb{R} \cap [0,1]$ – and $|\phi_0\rangle$ denotes the system's initial state, we obtain the solution

$$|\phi_t\rangle = \exp(t\boldsymbol{H})|\phi_0\rangle. \tag{2}$$

Accordingly, the *unitary operator* $\exp(t\boldsymbol{H}) =: \boldsymbol{U}_t$ determines the system's *time evolution*.

The implementation of quantum computing in the gate-circuit model, c.f. Figure 1, decomposes into two parts (see Section II for a formal introduction to the terminology and notation we employ in the following). In the *quantum part*, the computer successively applies a *gate sequence*

$$\underline{\boldsymbol{u}}(n_1), \ldots, \underline{\boldsymbol{u}}(n_M) \in \mathrm{U}(\mathcal{H}), \quad M \in \mathbb{N},$$

to a state vector $|\phi_0\rangle$, where the sequence's members belong to a finite and fixed *gate family*

$$\underline{\boldsymbol{u}} \in \underbrace{\big\{\underline{\boldsymbol{v}} : \{0, \ldots, N\} \to \mathrm{U}(\mathcal{H}),\, \underline{\boldsymbol{v}}(0) = \mathrm{Id}\big\}}_{=:\mathrm{U}_N(\mathcal{H})},\ N \in \mathbb{N}.$$

The application results in a time evolution of the form (1) that, for all discrete "computational" instants of time $T, \ldots, MT$, $T := 1/M$, satisfies

$$|\phi_{mT}\rangle = \underline{\boldsymbol{u}}(n_m) \cdots \underline{\boldsymbol{u}}(n_1)|\phi_0\rangle, \quad m \in \{1, \ldots, M\}.$$

The *classical part*, which is processed by classical digital hardware, *controls* the application of the gate sequence based on inputs $\boldsymbol{U} \in \mathrm{U}(\mathcal{H})$ and $L \in \mathbb{N}$. Particularly, given $L \in \mathbb{N}$, it consists of computing $\boldsymbol{n} := (n_1, \ldots, n_M)$, such that for $\underline{\boldsymbol{u}}(\boldsymbol{n}) := \underline{\boldsymbol{u}}(n_m) \cdots \underline{\boldsymbol{u}}(n_1)$ and all initial states $|\phi_0\rangle$, we have

$$\big\|\boldsymbol{U}|\phi_0\rangle - |\phi_{MT}\rangle\big\| = \big\|\big(\boldsymbol{U} - \underline{\boldsymbol{u}}(\boldsymbol{n})\big)|\phi_0\rangle\big\| < \frac{1}{2^L}$$

In other words, gate-circuit quantum computing ultimately aims to *approximate* the action of some unitary operator $\boldsymbol{U}$ up to a desired accuracy of $1/2^L$. If there exists a suitable gate sequence for every unitary operator $\boldsymbol{U}$ and every $L \in \mathbb{N}$, we call the gate family $\underline{\boldsymbol{u}} \in \mathrm{U}_N(\mathcal{H})$ *universal*.

In abstract mathematical terms, the universality of a gate family $\underline{\boldsymbol{u}} \in \mathrm{U}_N(\mathcal{H})$, $N \in \mathbb{N}$, guarantees the existence of a mapping

$$\mathtt{QC} : \mathrm{U}(\mathcal{H}) \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}}), \tag{3}$$

where $\mathrm{G}(\underline{\boldsymbol{u}})$ is the *monoid* generated by $\underline{\boldsymbol{u}}$, such that for every unitary operator $\boldsymbol{U} \in \mathrm{U}(\mathcal{H})$ and every accuracy $1/2^L$, $L \in \mathbb{N}$, we have

$$\|\boldsymbol{U} - \mathtt{QC}(\boldsymbol{U}, N)\| < \frac{1}{2^L}.$$

For the purpose of unambiguous terminology, we will refer to $\mathtt{QC}$ as a *quantum compiler function*, and to the problem of implementing $\mathtt{QC}$ on some hardware platform as the *quantum compiler problem* throughout the remainder of this article.

As indicated above, quantum compiling is an entirely classical task. The process of applying the physical quantum gates is controlled by some digital algorithm, a principle mirrored in emerging quantum programming languages [7]. Particularly, this algorithm has to evaluate the relevant quantum compiler function for the given input. There exist various analytic results on quantum compiler functions, the arguably most famous one being the *Solovay-Kitaev theorem*, c.f. [8], [9], [10]. Contemporary research largely focuses on optimizing quantum compiler functions with regards to different performance metrics and specific purposes, including but not limited to minimizing the count of required ancilla qubits [11] or certain quantum gates [12], [13]. Advanced and interdisciplinary topics in gate-based quantum compiling and computing concern, for example, distributed information processing architectures [14], [15] or the *entirely* digital simulation of quantum algorithms [16], [17]. Yet, the design of gate-based implementations of quantum algorithms remains challenging, despite the comprehensive research efforts [7].

Throughout the relevant literature, the analysis and optimization of quantum compiler functions generally concern the respective function at an *analytical* level; i.e., as an (abstract) mathematical object, potentially including its *algebraic* computation. Albeit not mentioned explicitly, this presumes any actual algorithmic implementation of such a function to depend – in terms of its performance and desired properties – *only* on the (abstract) function itself, but *not* the specific algorithmic implementation or properties of the underlying digital hardware. For example, if some quantum compiler function promises to minimize the count of a specific gate, any algorithmic implementation of that function should provide this minimization, regardless of

- which specific method is chosen to implement and compute the function, provided the method *does* implement the function faithfully;
- the physical specifications of the digital hardware executing the algorithm.

However, we can guarantee this form of "hardware and implementation independence" *only* if the function exhibits a faithful algorithmic implementation (in a mathematically strict sense) in the first place. Through this article, we address the question of whether this requirement can be fulfilled.

In a previous work, we have considered the related problem of whether there exist faithful algorithmic implementations of *quantum gate-circuit emulation functions* [18]. Instead of approximating a single target unitary transformation, quantum gate-circuit emulation functions serve the purpose of "rewriting" a quantum circuit that uses gates from some input gate family into an equivalent (up to a prescribed error tolerance) quantum circuit that uses gates from a fixed output gate family. Informally, we can think of this problem as a quantum analogon to the problem of expressing a classical boolean function through different *functionally complete* sets of digital logic gates. Using the framework of computable analysis, we have shown that much in contrast to its classical counterpart, the problem of quantum gate-circuit emulation is not solvable by means of digital algorithms. In the present work, we extend this analysis to quantum compiler functions, and show that they do not exhibit faithful algorithmic implementations either. The remainder of this article is structured as follows. Section II provides a concise review of quantum compiler functions as considered in the relevant literature and substantiates the engineering problem addressed in this article in mathematical terms. Section III provides a concise introduction to the required mathematics of computable analysis and establishes the framework of unitary computable operators underlying our theory. Subsequently, Section IV presents our main result. We prove that digital quantum compiling in a mathematically rigorous sense is infeasible. Section V discusses the implications of this result for (heuristic) digital implementations of quantum compiler functions and closes the paper with a brief general subsumption of our analysis.

A. Notation, Terminology, and General Remarks

Throughout the remainder of this article, the letters n, N, m, M, l, L, k, K, j, and J denote *generic* natural-number variables. Especially, we do *not* reserve any of them for specific quantities; their purpose is always with respect to and unambiguous from the relevant context. In some cases, for example (c.f. Section II), N will refer to the *dimension* of a *Hilbert space*, while in other cases, it will refer to the *number of members* of some *gate family*.

For generic sets $\mathcal{A}$ and $\mathcal{B}$, a *partial function* $f : \mathcal{A} \supseteq\!\!\rightarrow \mathcal{B}$ is of the form $f : D(f) \to \mathcal{B}$, $D(f) \subseteq \mathcal{A}$. We call $D(f)$ the *domain* of f. If $D(f) = \mathcal{A}$, we call f a *total* function. Since the inclusion $D(f) \subseteq \mathcal{A}$ includes the *improper* case, every total function is also partial, but not vice versa.

Consider $N \in \mathbb{N}$ and generic functions $f_1 : \mathcal{A}_1 \to \mathcal{A}_2, \ldots \ldots, f_N : \mathcal{A}_N \to \mathcal{A}_{N+1}$. For $a \in \mathcal{A}_1$, we define

$$[f_N \cdots f_1](a) := f_N(\cdots f_1(a) \cdots),$$

i.e., $[f_N \cdots f_1] : \mathcal{A}_1 \to \mathcal{A}_{N+1}$ is the concatenation of the functions $f_1, \ldots, f_N$. Note that aside from indicating the concatenation of functions, *square brackets* will fulfill additional (common) notational purposes (such as denoting, e.g., *intervals*). The relevant meaning will always be unambiguous from the respective context.

If an expression $\mathrm{expr}(\cdot)$ of some form defines the elements of a sequence $(a_n)_{n\in\mathbb{N}}$, we employ the notation

$$(a_n)_{n\in\mathbb{N}} : a_n = \mathrm{expr}(n)$$

for the sequence's definition. For example, for $\mathrm{expr}(\cdot) \equiv \sqrt{\cdot}$, we may write $(a_n)_{n\in\mathbb{N}} : a_n = \sqrt{n}$.

II. The Quantum Compiler Problem

Throughout this article, we consider an arbitrary but fixed N-dimensional Hilbert space over $\mathbb{C}$,

$$\mathcal{H} := \mathrm{span}\big\{|\boldsymbol{e}_1\rangle, \ldots, |\boldsymbol{e}_N\rangle\big\}, \qquad N \in \mathbb{N},\ N \geq 2.$$

The *computational orthonormal basis* $|\boldsymbol{e}_1\rangle, \ldots, |\boldsymbol{e}_N\rangle$ induces a global isometry between linear operators $\boldsymbol{A} : \mathcal{H} \to \mathcal{H}$ and complex-valued N-by-N matrices. We will thus identify each such operator with its corresponding element of $\mathbb{C}^{N\times N}$ in the usual way. Given $\boldsymbol{A} \in \mathbb{C}^{N\times N}$, we denote by $\boldsymbol{A}^\dagger$ the *Hermitian adjoint* and by $\boldsymbol{A}^{-1}$, if it exists, the *inverse* of $\boldsymbol{A}$. A matrix $\boldsymbol{H} \in \mathbb{C}^{N\times N}$ is *skew-Hermitian* if it satisfies $\boldsymbol{H}^\dagger = -\boldsymbol{H}$. Furthermore, we call

- $\mathrm{U}(N) := \{\boldsymbol{U} \in \mathbb{C}^{N\times N} : \boldsymbol{U}^\dagger = \boldsymbol{U}^{-1}\}$ the *general unitary group* of degree N;
- $\mathrm{SU}(N) := \{\boldsymbol{U} \in \mathrm{U}(N) : \det \boldsymbol{U} = 1\}$ the *special unitary group* of degree N;
- $\boldsymbol{U} \in \mathrm{U}(N)$ a *unitary matrix*.

Since our theory equally applies to the general and special unitary group of *any* degree $N \geq 2$, we will abbreviate $\mathrm{U}(N) \equiv \mathrm{U}$ and $\mathrm{SU}(N) \equiv \mathrm{SU}$ unless it leads to ambiguities.

Given any skew-Hermitian matrix $\boldsymbol{H} \in \mathbb{C}^{N\times N}$, $N \in \mathbb{N}$, we have $\exp(\boldsymbol{H}) \in \mathrm{U}(N)$. Further, if $\boldsymbol{H}$ satisfies $\mathrm{tr}H = 0$ – that is, if $\boldsymbol{H}$ is *trace free* – we have $\exp(\boldsymbol{H}) \in \mathrm{SU}(N)$. More generally, we have

$$\exp(\mathrm{tr}\boldsymbol{H}) = \det \exp(\boldsymbol{H})$$

which, in the context of quantum physics, characterizes a quantum system's *global phase*. Notably, the global phase is *not* an *observable*, i.e., it cannot be measured. Given any $\theta \in \mathbb{R}$, the Hamiltonian operators $\boldsymbol{H}$ and $\boldsymbol{H} + \jmath\theta \cdot \mathrm{Id}$ are physically *indistinguishable*. Throughout the literature of quantum computing, it is thus customary to consider the special unitary group to begin with. Unless an explicit distinction between the special and the general unitary group is necessary, we will follow this convention hereafter. However, we emphasize that *all* of this paper's theoretical results are equally valid for the special *and* the general unitary group.

Consider $N \in \mathbb{N}$ arbitrary. As indicated in Section I, a *gate family* is a mapping $\underline{\boldsymbol{u}} : \{0, \dots, N\} \to \mathrm{SU}$ such that we have $\underline{\boldsymbol{u}}(0) = \mathrm{Id}$. We denote the set of all such mappings by SU_N. Given $\underline{\boldsymbol{u}} \in \mathrm{SU}_N$, a tuple

$$\boldsymbol{n} = (n_1, \dots, n_m) \in \{0, \dots, N\}^M, \quad M \in \mathbb{N},$$

denotes a *symbolic* quantum circuit. Since we require $\underline{\boldsymbol{u}}$ to satisfy $\underline{\boldsymbol{u}}(0) = \mathrm{Id}$, we may consider $\boldsymbol{n} \in \{0, \dots, N\}^M$ such that $n_m \neq 0$. In contrast to the symbolic nature of $\boldsymbol{n}$, the corresponding unitary matrix

$$\underline{\boldsymbol{u}}(\boldsymbol{n}) := \underline{\boldsymbol{u}}(n_m) \cdots \underline{\boldsymbol{u}}(n_1) \in \mathrm{SU}$$

is an abstract mathematical object that characterizes a quantum-physical state transformation. The relevant literature refers to $\underline{\boldsymbol{u}}(\boldsymbol{n})$ as a *matrix representation* [19, Section 2.1.2, p. 63 ff]. Often, the distinction between both cases is not explicit. For a thorough analysis of computability, it is, however, necessary to differentiate symbolic quantum circuits from matrix representations.

Observe that for all $N \in \mathbb{N}$, there exists a (unique) function $\mathrm{bx}_N : \mathbb{N} \times \mathbb{N} \to \{0, \dots, N\}$ that satisfies

$$n = \sum_{m=0}^{\infty} \mathrm{bx}_N(n, m) \cdot (N+1)^m.$$

for all $n \in \mathbb{N}$. In other words, bx_N provides the base-$(N+1)$ expansion of any number $n \in \mathbb{N}$. Then, given $\underline{\boldsymbol{u}} \in \mathrm{SU}_N$ as above, we call

$$\mathrm{G}(\underline{\boldsymbol{u}}) := \left\{ \prod_{\infty}^{m=0} \underline{\boldsymbol{u}}(\mathrm{bx}_N(n, m)) : n \in \mathbb{N} \right\}$$

(where we use the inverted placement of "$m = 0$" and "∞" to indicate a *right-to-left* product) the *monoid* generated by $\underline{\boldsymbol{u}}$. In other words, $\mathrm{G}(\underline{\boldsymbol{u}})$ equals the set of exactly those unitary matrices that correspond to some symbolic quantum circuit (with respect to the underlying gate family $\underline{\boldsymbol{u}}$). For $\boldsymbol{U} \in \mathrm{SU}$, $L \in \mathbb{N}$, we define

$$\Delta_{\mathrm{G}(\underline{\boldsymbol{u}})}(\boldsymbol{U}, L) := \big\{ \boldsymbol{V} \in \mathrm{G}(\underline{\boldsymbol{u}}) : \|\boldsymbol{U} - \boldsymbol{V}\| < 2^{-L} \big\}.$$

Then, the gate family $\underline{\boldsymbol{u}}$ is *universal* if and only if we have $\Delta_{\mathrm{G}(\underline{\boldsymbol{u}})}(\boldsymbol{U}, L) \neq \emptyset$ for *all* $\boldsymbol{U} \in \mathrm{SU}$, $L \in \mathbb{N}$ – that is, if and only if $\mathrm{G}(\underline{\boldsymbol{u}})$ is (topologically) *dense* in SU.

In the context of quantum computing, we may consider the Solovay-Kitaev theorem primarily an abstract mathematical statement about *circuit complexity*. To the best of the authors' knowledge, the first attempt at leveraging the theorem's proof for the construction of an explicit quantum compiler function is due to *C. M. Dawson* and *M. A. Nielsen* [9]. For $\epsilon > 0$ sufficiently small and suitable gate families $\underline{\boldsymbol{u}} \in \mathrm{SU}_N$, $N \in \mathbb{N}$, they base their approach on a mapping $\mathtt{SK}_0 : \mathrm{SU} \to \{\underline{\boldsymbol{u}}(0), \dots, \underline{\boldsymbol{u}}(N)\}$ that satisfies

$$\|\boldsymbol{U} - \mathtt{SK}_0(\boldsymbol{U})\| < \epsilon$$

for all $\boldsymbol{U} \in \mathrm{SU}$ – that is, $\mathtt{SK}_0$ provides an initial *base-circuit* approximation to $\boldsymbol{U}$ –, as well as *approximate balanced group commutators* $[\cdot]_{\tilde{\mathrm{L}}}, [\cdot]_{\tilde{\mathrm{R}}} : \mathrm{SU} \to \mathrm{SU}$. Setting

$$[\boldsymbol{U}, m]_{\mathrm{L}} := \mathtt{SK}\big(\big[\boldsymbol{U} \mathtt{SK}^{\dagger}(\boldsymbol{U}, m) \big]_{\tilde{\mathrm{L}}}, m \big),$$

$$[\boldsymbol{U}, m]_{\mathrm{R}} := \mathtt{SK}\big(\big[\boldsymbol{U} \mathtt{SK}^{\dagger}(\boldsymbol{U}, m) \big]_{\tilde{\mathrm{R}}}, m \big),$$

for $m \in \mathbb{N}$, they successively continue to improve the approximation, recursively defining $\mathtt{SK} : \mathrm{SU} \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$ through

$$\begin{aligned} \mathtt{SK}(\boldsymbol{U}, 0) &:= \mathtt{SK}_0(\boldsymbol{U}), \\ \mathtt{SK}(\boldsymbol{U}, m+1) &:= \dots \\ &\dots [\boldsymbol{U}, m]_{\mathrm{L}} [\boldsymbol{U}, m]_{\mathrm{R}} [\boldsymbol{U}, m]_{\mathrm{L}}^{\dagger} [\boldsymbol{U}, m]_{\mathrm{R}}^{\dagger} \mathtt{SK}(\boldsymbol{U}, m). \end{aligned} \tag{4}$$

Subsequently, they sketch the existence of a *polylogarithmic* function $\xi_{\mathtt{SK}} : \mathbb{N} \to \mathbb{R}$ such that for all $m, M \in \mathbb{N}$, $m \geq \xi_{\mathtt{SK}}$,we have $\|\boldsymbol{U} - \mathtt{SK}(\boldsymbol{U}, m)\| < 1/2^M$. Accordingly, defining $\mathtt{QC}_{\mathtt{SK}} : \mathrm{SU} \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$ through

$$\mathtt{QC}_{\mathtt{SK}}(\boldsymbol{U}, M) := \mathtt{SK}\big(\boldsymbol{U}, \min\{ m \in \mathbb{N} : m \geq \xi_{\mathtt{SK}}(M) \} \big)$$

provides a quantum compiler function in the sense of Section I. As follows from (4), the unitary matrix $\mathtt{SK}(\boldsymbol{U}, m)$ corresponds to some symbolic quantum circuit of length at most 5^m. Accordingly, since $\xi_{\mathtt{SK}}$ is polylogarithmic, the circuit length required to approximate $\boldsymbol{U}$ up to an accuracy of m binary digits grows *polynomially* in m. A less favorable scaling behavior would likely render quantum circuits unusable for practical applications. In this sense, the Solovay-Kitaev theorem is a theoretical prerequisite for gate-based quantum computing.

Even before *C. M. Dawson's* and *M. A. Nielsen's* characterization of $\mathtt{QC}_{\mathtt{SK}}$, [20] determined the optimal circuit-length scaling behavior any quantum compiler function may provide. Given $\underline{\boldsymbol{u}} \in \mathrm{SU}_N$, $N \in \mathbb{N}$, and $\boldsymbol{U} \in \mathrm{G}(\underline{\boldsymbol{u}})$, we denote the *smallest* $l \in \mathbb{N}$ such that for some $n \in \mathbb{N}$ satisfying $n < (N+1)^l$, we have

$$\boldsymbol{U} = \prod_{\infty}^{m=0} \underline{\boldsymbol{u}}(\mathrm{bx}_N(n, m))$$

by $|\boldsymbol{U}|$. That is, $|\boldsymbol{U}|$ denotes the *word length* of $\boldsymbol{U}$. For $l \in \mathbb{N}$ arbitrary, we furthermore denote the set of unitary matrices $\boldsymbol{U} \in \mathrm{G}(\underline{\boldsymbol{u}})$ whose word length is smaller than or equal to l by $\mathrm{G}_{|\cdot| \leq l}(\underline{\boldsymbol{u}})$, i.e.,

$$\mathrm{G}_{|\cdot| \leq l}(\underline{\boldsymbol{u}}) := \big\{ \boldsymbol{V} \in \mathrm{G}(\underline{\boldsymbol{u}}) : |\boldsymbol{V}| \leq l \big\} \subset \mathrm{G}(\underline{\boldsymbol{u}}).$$

Then, assuming $\underline{\boldsymbol{u}}$ exhibits a *spectral gap* [20], there exists numbers $k_{\underline{\boldsymbol{u}}}, j_{\underline{\boldsymbol{u}}} \in \mathbb{N}$ such that for all $l, L \in \mathbb{N}$ satisfying $l \geq k_{\underline{\boldsymbol{u}}} \cdot L + j_{\underline{\boldsymbol{u}}}$, we have

$$\Delta_{\mathrm{G}(\underline{\boldsymbol{u}})}(\boldsymbol{U}, L) \cap \mathrm{G}_{|\cdot| \leq l}(\underline{\boldsymbol{u}}) \neq \emptyset.$$

From an analytic point of view, the axiom of choice then implies the existence of quantum compiler functions $\mathtt{QC} : \mathrm{SU} \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$ that, for all such l, L, satisfy

$$\mathtt{QC}(\boldsymbol{U}, L) \in \Delta_{\mathrm{G}(\underline{\boldsymbol{u}})}(\boldsymbol{U}, L) \cap \mathrm{G}_{|\cdot| \leq l}(\underline{\boldsymbol{u}}).$$

Yet, the mere mathematical existence of such functions does not imply their realizability in terms of an algorithm.

The attainability of linear circuit-length scaling through suitable quantum compiler functions is a *nonconstructive* result. Contrary to numerous claims of the opposite, the same holds true for the existence of $\mathtt{QC}_{\mathtt{SK}}$: Computing $\mathtt{SK}_0$ – recall that $\mathtt{SK}_0$ is fundamental to the definition of $\mathtt{QC}_{\mathtt{SK}}$ – is beyond of

what constructive mathematics considers feasible.[1] First and foremost, this concerns the ability to perform arithmetical operations of *infinite* precision. Interestingly, *C. M. Dawson* and *M. A. Nielsen* themselves note that

> *"A caveat to* [our] *discussion is that we have entirely ignored the* precision *with which arithmetical operations are carried out. In practice we cannot compute with precisely specified unitary operations,* […] *since such operations in general require an infinite amount of classical information to specify. Instead, we must work with approximations to such operations, and* [must] *do all our calculations using finite precision arithmetic. A complete analysis of the SK algorithm should include an account of the cost of such approximations."*

They continue to explain that "[. . .] *there seems to be little need for such an analysis*," since their implementation in standard floating-point arithmetics provided sufficient results given the error constraints they considered. Nevertheless, the assumption of infinite precision places the practical applicability of their analysis on a heuristic foundation.

In order to obtain a constructive theory of digital algorithms for quantum compiling, it is insufficient to merely analyze quantum compiler functions on an abstract level. Instead, we must consider the relevant quantum compiler functions in the context of a constructive model of digital computing. Throughout the community of computer science, Turing machines and, drawing thereupon, *computable (effective) analysis* are widely accepted as the most suitable model for this purpose, providing the tools to formalize and investigate the quantum compiler problem anticipated in Section I.

Quantum Compiler Problem

Given $N \in \mathbb{N}$, $\underline{\boldsymbol{u}} \in \mathrm{U}_N(\mathcal{H})$, and any quantum compiler function $\mathtt{QC} : \mathrm{U}(\mathcal{H}) \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$, find an algorithm that *faithfully* implements $\mathtt{QC}$. That is, such that the algorithm

- provably preserves the function's analytic properties;
- allows for proving the implementation's correctness by methods of formal verification.

From the perspective of computable analysis, the quantum compiler problem asks for a *realizer* of the relevant quantum compiler function. If such a realizer exists, we call the quantum compiler function *Turing computable*. In the subsequent Section III, we will formally introduce the concept of Turing computability and restate the quantum compiler problem accordingly. This paper's core contribution consists of proving the quantum compiler problem *infeasible*. Due to the unitary group's topological properties, quantum compiler functions in the sense of Section I turn out to be necessarily *uncomputable*. As a consequence of our results, there cannot exist a constructive proof of the existence of a quantum compiler function, which we will discuss in detail in Section IV-A. Accordingly, a rethinking of quantum compiling as currently considered in the relevant literature is necessary.

[1] Although we have yet to verify the details, we suspect the same is the case for computing the balanced group commutators $[\cdot]_{\tilde{\mathrm{L}}}$ and $[\cdot]_{\tilde{\mathrm{R}}}$.

III. Preliminaries: Turing Machines, Effective Analysis, and the Computable Unitary Group

Throughout this section, we introduce the theory of *computable unitary matrices*, drawing upon the framework of *effective analysis*. For a comprehensive treatment of *computability theory*, *computable analysis*, and *constructive mathematics*, we refer the reader to, e.g., [21], [22], [23], [24], [25].

As indicated in Section II, effective analysis builds upon the work of *Alan M. Turing* [26], [27]. The *Turing machine*, which forms the core concept of Turing's computability theory, provides an idealized but definitive model of digital computers. Hence, if we cannot theoretically implement some computational procedure on a Turing machine, we can definitely not implement it on a real digital computer. in practice. From Section I, recall that quantum compiling relies on digital hardware in almost all contemporary quantum computers. Thus, the analysis of algorithms for quantum compiling necessarily falls into the domain of Turing's theory.

Turing machines are state-based automata that systematically alter the content of a memory tape. While they ultimately form the basis of effective analysis, it is – due to the necessity of computing functions of natural numbers – advantageous to approach computability from a slightly different direction. We consider the set of μ-*recursive functions*, which is the *smallest* possible set that

- consists of *partial functions* $g : \mathbb{N} \times \cdots \times \mathbb{N} \supseteq\!\!\to \mathbb{N}$;
- contains the *successor function*, all *constant functions*, and all *projection functions* [21, Definition 2.1, p. 8f];
- is closed under the operations *composition*, *primitive recursion*, and *unbounded search* [21, Definition 2.1, p. 8f, Definition 2.2, p. 10].

Notably, Turing machines and μ-recursive functions are *equivalent* as a model of computability. In other words, a function $g : \mathbb{N} \times \cdots \times \mathbb{N} \supseteq\!\!\to \mathbb{N}$ is μ-recursive if and only if there exists a Turing machine that computes g [28].

The partiality of μ-recursive functions exhibits a dedicated interpretation. Consider a Turing machine that computes some μ-recursive function $g : \mathbb{N} \times \cdots \times \mathbb{N} \supseteq\!\!\to \mathbb{N}$ and recall that Turing machines are state-based automata that execute computations in sequential processing steps. The function's domain $D(g) \subseteq \mathbb{N} \times \cdots \times \mathbb{N}$ corresponds to the set of inputs for which the Turing machine eventually terminates its computation and returns some (well-defined) output. As a consequence of the well-known *halting problem*, $D(g)$ is, in general, *not* a *recursive* subset of $\mathbb{N} \times \cdots \times \mathbb{N}$. That is, the *indicator function*

$$\mathbb{1}_{D(g)} : \mathbb{N} \times \cdots \times \mathbb{N} \to \{0,1\},\ n \mapsto \begin{cases} 1 & \text{if } \boldsymbol{n} \in D(g), \\ 0 & \text{otherwise,} \end{cases}$$

is *not* necessarily a μ-recursive function. Once we have established a formal notion of computability for some abstract sets $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$, we may attempt to prove the uncomputability of a given function $F : D(F) \to \mathcal{B}$, $D(F) \subseteq \mathcal{A}_\mu$, by reducing the halting problem to the evaluation of F for a suitable sequence of argument values. In the final analysis, the results we establish throughout this article are in turn a consequence of the halting problem.

As emphasized on several occasions, a precise notion of computability for quantum compiler functions is indispensable for answering the quantum compiler problem in a mathematically satisfactory manner. Evidently, we cannot directly apply μ-recursive functions to unitary matrices. Accordingly, any theory of algorithms for quantum compiling must address the question of how to represent unitary matrices within the realm on natural numbers. This conforms to our intuitive understanding of real-world digital computers: On the lowest level, we must represent any object we wish to perform computations with by a bit-string, i.e., the base-two expansion of some natural number. If the object is of abstract analytical nature – as is the case for unitary matrices –, we require additional information on how to "interpret" the bit-string. This principle is formalized by the notion of *numberings*.

Given a countable set $\mathcal{A}_\mu$, any mathematically well-defined partial surjective function $\mathsf{N}_\mathcal{A} : \mathbb{N} \supseteq\!\!\rightarrow \mathcal{A}_\mu$ defines a numbering of $\mathcal{A}_\mu$. Thus, $\mathsf{N}_\mathcal{A}$ is primarily an abstract mathematical object itself and, from the perspective of computable analysis, possesses no intrinsic meaning. Rather, its meaning is entirely *operational*: The numbering $\mathsf{N}_\mathcal{A}$ determines which operations that involve the elements of $\mathcal{A}_\mu$ are computable. Below, we will provide formal definitions of *computable functions* and *computable sequences*. We will then start defining and analyzing explicit numberings for the *computable real numbers*, the *computable complex numbers*, and, ultimately, the *computable unitary matrices*. Subsequently, we require some additional technicalities. Throughout the remainder of this article, we fix

- an *admissible enumeration* $(\breve{\boldsymbol{\varphi}}_n)_{n\in\mathbb{N}}$ of μ-recursive functions [21, Chapter I, Section 3, p. 14ff],
- a μ-recursive *pairing function* $\mathbf{p} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, and the projections $\mathbf{f}, \mathbf{s} : \mathbb{N} \to \mathbb{N}$ corresponding to the pairing function's inverse [23, Section 1.4, p. 12].

Given any numbered sets $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$, we may canonically define the *product numbering* $\mathsf{N}_{\mathcal{A}\times\mathcal{B}} : \mathbb{N} \supseteq\!\!\rightarrow \mathcal{A}_\mu \times \mathcal{B}_\mu$ of $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ through

$$\mathsf{N}_{\mathcal{A}\times\mathcal{B}}(n) := \big([\mathsf{N}_\mathcal{A}\mathbf{f}](n), [\mathsf{N}_\mathcal{B}\mathbf{s}](n)\big),$$

where $D(\mathsf{N}_{\mathcal{A}\times\mathcal{B}}) := \{\mathbf{p}(m,l) : (m,l) \in D(\mathsf{N}_\mathcal{A}) \times D(\mathsf{N}_\mathcal{B})\}$. Observe that for $N \in \mathbb{N}$, $N \geq 3$, the sets $\mathcal{A}_\mu^N$ and $\mathcal{A}_\mu^{N\times N}$ are *nested products*, i.e., we have

$$\mathcal{A}_\mu^N = \mathcal{A}_\mu \times \mathcal{A}_\mu^{N-1} \quad \text{and} \quad \mathcal{A}_\mu^{N\times N} = \mathcal{A}_\mu^N \times \mathcal{A}_\mu^{(N-1)\times N}.$$

Accordingly, the product numbering inductively provides a numbering for the set $\mathcal{A}_\mu^{N\times N}$, which we will refer to as the (canonic) *matrix numbering* of $\mathcal{A}_\mu^{N\times N}$ further below.

Definition 1 (Computable Functions and Sequences):

- Consider numberings $\mathsf{N}_\mathcal{A} : \mathbb{N} \supseteq\!\!\rightarrow \mathcal{A}_\mu$ and $\mathsf{N}_\mathcal{B} : \mathbb{N} \supseteq\!\!\rightarrow \mathcal{B}_\mu$ and let $F : \mathcal{A}_\mu \supseteq\!\!\rightarrow \mathcal{B}_\mu$ be any partial function. Assume there exists a μ -recursive function $g : \mathbb{N} \supseteq\!\!\rightarrow \mathbb{N}$ s.t. for all $m \in \{n \in D(\mathsf{N}_\mathcal{A}) | \mathsf{N}_\mathcal{A}(n) \in D(F)\}$, we have
$$n \in D(g) \quad \text{and} \quad [F\mathsf{N}_\mathcal{A}](n) = [\mathsf{N}_\mathcal{B} g](n).$$
We call F an $(\mathsf{N}_\mathcal{A}, \mathsf{N}_\mathcal{B})$-computable function.
- Consider a numbering $\mathsf{N}_\mathcal{A} : \mathbb{N} \supseteq\!\!\rightarrow \mathcal{A}_\mu$ and let $h : \mathbb{N} \to \mathbb{N}$ be any unary total μ-recursive function. We call
$$(a_n)_{n\in\mathbb{N}} : a_n = [\mathsf{N}_\mathcal{A} h](n)$$
a onefold $\mathsf{N}_\mathcal{A}$-computable sequence. For $N \in \mathbb{N}$, any N-ary total μ-recursive function $h : \mathbb{N} \times \cdots \times \mathbb{N} \to \mathbb{N}$ analogously defines an N-fold $\mathsf{N}_\mathcal{A}$-computable sequence.

If, given numbered sets $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$, the relevant numberings $\mathsf{N}_\mathcal{A}$ and $\mathsf{N}_\mathcal{B}$ are unambiguous from the context, we refer to $(\mathsf{N}_\mathcal{A}, \mathsf{N}_\mathcal{B})$-computable functions merely as *computable*. Furthermore, for $N \in \mathbb{N}$, we refer to any N-fold $\mathsf{N}_\mathcal{A}$-computable sequence merely as a *computable sequence* in this case, unless mentioning the number of sequence indices or underlying function's arity is strictly necessary.

Consider any numbered set $\mathcal{A}_\mu$ and let $d_\mathcal{A} : \mathcal{A}_\mu \times \mathcal{A}_\mu \to \{x \in \mathbb{R} : x \geq 0\}$ a *metric*. Further, assume the following for all computable sequences $(a_{n,m})_{n,m\in\mathbb{N}} \subset \mathcal{A}_\mu$:

- If there exists a μ-recursive function $\xi : \mathbb{N} \to \mathbb{N}$ such that for all $n, l, k, M \in \mathbb{N}$, we have
$$d_\mathcal{A}\big(a_{n,\xi(M)+l}, a_{n,\xi(M)+k}\big) < \frac{1}{2^M},$$
then $(a_{n,*})_{n\in\mathbb{N}} : a_{n,*} = \lim_{m\to\infty} a_{n,m}$ defines a computable sequence as well.

We call $\mathcal{A}_\mu$ *effectively closed*, and $(a_{n,*})_{n\in\mathbb{N}}$ the *effective limit* of $(a_{n,m})_{n,m\in\mathbb{N}}$ for $m \to \infty$. Consider the *(effective) limit operator*

$$\lim\nolimits_\mathcal{A} : \ddot{\mathcal{A}}_\mu \supseteq\!\!\rightarrow \mathcal{A}_\mu,\ (a_n)_{n\in\mathbb{N}} \mapsto a_* := \lim\nolimits_{n\to\infty} a_n,$$

where $\ddot{\mathcal{A}}_\mu$ denotes the set of onefold $\mathsf{N}_\mathcal{A}$-computable sequences and we have $(a_n)_{n\in\mathbb{N}} \in D(\lim_\mathcal{A})$ if and only if the effective limit of $(a_n)_{n\in\mathbb{N}}$ exists. Furthermore, assume there exists a μ-recursive function $g : \mathbb{N} \times \mathbb{N} \supseteq\!\!\rightarrow \mathbb{N}$ that satisfies the following for all $n, m \in \mathbb{N}$:

- Whenever $\breve{\boldsymbol{\varphi}}_n$ and $\breve{\boldsymbol{\varphi}}_m \equiv \xi$ are total unary μ-recursive functions such that for all $l, k, M \in \mathbb{N}$, we have
$$d_\mathcal{A}\Big([\mathsf{N}_\mathcal{A}\breve{\boldsymbol{\varphi}}_n](\xi(M)+l), [\mathsf{N}_\mathcal{A}\breve{\boldsymbol{\varphi}}_n](\xi(M)+k)\Big) < \frac{1}{2^M},$$
we also have $[\mathsf{N}_\mathcal{A} g](n,m) = \lim_{M\to\infty} [\mathsf{N}_\mathcal{A}\breve{\boldsymbol{\varphi}}_n](M)$.

We call the limit operator $\lim_\mathcal{A}$ $\mathsf{N}_\mathcal{A}$-computable. In other words, the computability of the relevant limit operator is a (mathematically) stronger version of a numbered set's effective closedness. Computability of limit operators will play a minor role throughout this article. Primarily, it will be part of our substantiation of the computable general and special unitary group's definition below.

The idea of *Turing* computable functions, as formalized in Definition 1, conforms to our intuitive understanding of algorithms – computational procedures that accept inputs of some type and return outputs of another one. Throughout this article, we will primarily concern ourselves with a different notion of function computability, which we will define and substantiate below.

Definition 2 (Banach-Mazur Computability of Functions): Let $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ be any numbered sets and consider a total function $F : \mathcal{A}_\mu \to \mathcal{B}_\mu$. If, for all $\mathsf{N}_\mathcal{A}$-computable sequences $(a_n)_{n\in\mathbb{N}}$, the sequence $(b_n)_{n\in\mathbb{N}} : b_n = F(a_n)$ is $\mathsf{N}_\mathcal{A}$-computable, we call the function F Banach-Mazur computable.

Rather than asking whether the function $F : \mathcal{A}_\mu \to \mathcal{B}_\mu$ is itself computable, Banach-Mazur computability concerns the

question of whether F *preserves* (sequential) computability – that is, whether given some algorithm that successively generates objects $a_0, a_1, a_2, \ldots \in \mathcal{A}_\mu$, there necessarily exist a another algorithm that successively generates the objects $F(a_0), F(a_1), F(a_2), \ldots \in \mathcal{B}_\mu$. To understand the relevance of this concept, observe that a large number of mathematical problems in quantum computing induce such algorithmic sequences. The induced sequence's index variable then corresponds to the the problem's different *instances*, and the objects themselves to unitary matrices. In Shor's algorithm for prime factorization, for example, each problem instance equals the respective number to be factorized. Consider some quantum compiler function that is – hypothetically – Banach-Mazur but *not* Turing computable and any mathematical problem of the above type. In this case, we at least know for sure that it is possible to find an algorithm that, for each instance of the problem at hand, computes the image of the respective unitary matrix under the quantum compiler function. Note that there indeed exist functions that are Banach-Mazur but *not* Turing computable (see, e.g., [29]).

Finally, consider numbered sets $\mathcal{A}_\mu$ and $\mathcal{B}_\mu$ once more and observe that the Turing computability of a function $F : \mathcal{A}_\mu \to \mathcal{B}_\mu$ implies the function's Banach-Mazur computability. Specifically, let $g_1 : \mathbb{N} \supseteq\!\to \mathbb{N}$ be any μ-recursive function that *realizes* the function F in the sense of Definition 1. Given any $\mathsf{N}_\mathcal{A}$-computable sequence $(a_n)_{n\in\mathbb{N}}$, there must exist a μ-recursive function $g_2 : \mathbb{N} \to \mathbb{N}$ such that for all $n \in \mathbb{N}$, we have $a_n = [\mathsf{N}_\mathcal{A} g_2](n)$ and thus $F(a_n) = [\mathsf{N}_\mathcal{B} g_1 g_2](n)$. But

$$h : \mathbb{N} \to \mathbb{N},\ n \mapsto h(n) := g_1(g_2(n)),$$

defines a μ-recursive function. Thus, the sequence $(b_n)_{n\in\mathbb{N}} : b_n = F(a_n)$ must be $\mathsf{N}_\mathcal{B}$-computable. Observe that accordingly, if F is *not* Banach-Mazur computable, it is *definitely not* Turing computable either.

Having established the relevant mathematical underpinnings, we are equipped to turn towards *computable unitary matrices*. We define the numbering $\mathsf{N}_\mathbb{Q} : \mathbb{N} \to \mathbb{Q}$ through

$$\mathsf{N}_\mathbb{Q}(n) := \frac{[\mathbf{ff}](n) - [\mathbf{sf}](n)}{1 + \mathbf{s}(n)}.$$

In Definitions 3 and 4 below, we will introduce *computable real* and *complex numbers* and, subsequently, the *computable general* and *special unitary group*.

Definition 3 (Computable Numbers):

- Consider and $x \in \mathbb{R}$ and assume there exist total μ-recursive functions $h, \xi : \mathbb{N} \to \mathbb{N}$ such that we have

$$\left|x - [\mathsf{N}_\mathbb{Q} h](n)\right| < \frac{1}{2^N} \quad \text{for all } n, N \in \mathbb{N},\ n \geq \xi(N). \tag{5}$$

  We call x a computable real number and denote the set of all such numbers by $\mathbb{R}_\mu$. Further, we define $\mathsf{N}_\mathbb{R} : \mathbb{N} \supseteq\!\to \mathbb{R}_\mu$ such that for $m \in \mathbb{N}$, $x \in \mathbb{R}_\mu$, we have $\mathsf{N}_\mathbb{R}(m) = x$ if and only if $\breve{\boldsymbol{\varphi}}_{\mathbf{f}(m)} \equiv h$ and $\breve{\boldsymbol{\varphi}}_{\mathbf{s}(m)} \equiv \xi$ satisfy (5).
- We call $z = x + \jmath y$, $x, y \in \mathbb{R}_\mu$, a computable complex number and denote the set of all such numbers by $\mathbb{C}_\mu$. Further, we define $\mathsf{N}_\mathbb{C} : \mathbb{N} \supseteq\!\to \mathbb{C}_\mu$ such that for $m \in \mathbb{N}$, $z \in \mathbb{C}_\mu$, we have $\mathsf{N}_\mathbb{C}(m) = z$ if and only if z and m satisfy $z = [\mathsf{N}_\mathbb{R}\mathbf{f}](m) + \jmath[\mathsf{N}_\mathbb{R}\mathbf{s}](m)$.

By $\mathsf{C}_{N\times N} : \mathbb{N} \supseteq\!\to \mathbb{C}_\mu^{N\times N}$, $N \in \mathbb{N}$, we denote the (canonic) matrix numbering of $\mathbb{C}_\mu^{N\times N}$ (see further above). Then, for $N \geq 2$, the definition of the computable variants of the general and special unitary group of degree $N \in \mathbb{N}$ is as follows.

Definition 4 (Computable Unitary Group):

- We call $\mathrm{U}_\mu(N) := \{\boldsymbol{U} \in \mathbb{C}_\mu^{N\times N} : \boldsymbol{U}^\dagger = \boldsymbol{U}^{-1}\}$ the computable general unitary group of degree $N \in \mathbb{N}$, $N \geq 1$, and define $\mathsf{N}_\mathrm{U} : \mathbb{N} \supseteq\!\to \mathrm{U}_\mu(N)$ through

$$\mathsf{N}_\mathrm{U}(n) := \mathsf{C}_{N\times N}(n) \quad \text{for } n \in D\big(\mathsf{N}_\mathrm{U}\big) := \ldots \\ \ldots \big\{n \in D(\mathsf{C}_{N\times N}) : \mathsf{C}_{N\times N}(n) \in \mathrm{U}(N)\big\}.$$

- We call $\mathrm{U}_\mu(N) := \{\boldsymbol{U} \in \mathrm{U}_\mu(N) : \det \boldsymbol{U} = 1\}$ the computable special unitary group of degree $N \in \mathbb{N}$, $N \geq 1$, and define $\mathsf{N}_\mathrm{SU} : \mathbb{N} \supseteq\!\to \mathrm{SU}_\mu(N)$ through

$$\mathsf{N}_\mathrm{SU}(n) := \mathsf{C}_{N\times N}(n) \quad \text{for } n \in D\big(\mathsf{N}_\mathrm{SU}\big) := \ldots \\ \ldots \big\{n \in D(\mathsf{C}_{N\times N}) : \mathsf{C}_{N\times N}(n) \in \mathrm{SU}(N)\big\}.$$

For $\boldsymbol{U} \in \mathrm{SU}_\mu$, consider $n \in D(\mathsf{N}_\mathrm{SU})$ such that we have $\mathsf{N}_\mathrm{SU}(n) = \boldsymbol{U}$. The number n is a *description* of $\boldsymbol{U}$, and the numbering N_SU determines how to *interpret* this description. From the above, recall that any (well-defined) partial surjective mapping $\mathsf{N}'_\mathrm{SU} : \mathbb{N} \supseteq\!\to \mathrm{SU}_\mu$ provides a numbering for SU_μ. Accordingly, N'_SU is *not* computationally meaningful per se, since a Turing machine cannot "understand" the abstract object N'_SU. In light of Definitions 1 and 2, the numbering N'_SU derives it's meaning entirely through the question of which operations are N'_SU-computable and which are not. Specifically, given the algebraic properties of the abstract set SU, we would naturally expect the following from a numbering of SU_μ.

1) The *group operation* $\mathrm{gop}_\mathrm{SU} : \mathrm{SU}_\mu \times \mathrm{SU}_\mu \to \mathrm{SU}_\mu$ defined through the matrix multiplication
$$\boldsymbol{U}\boldsymbol{V} =: \mathrm{gop}_\mathrm{SU}(\boldsymbol{U}, \boldsymbol{V})$$
shall be N'_SU-computable (with $D(\mathrm{gop}_\mathrm{SU}) = \mathrm{SU}_\mu \times \mathrm{SU}_\mu$).
2) The *metric* $d_\mathrm{SU} : \mathrm{SU}_\mu \times \mathrm{SU}_\mu \to \{x \in \mathbb{R}_\mu : x \geq 0\}$ defined through the matrix norm
$$\|\boldsymbol{U} - \boldsymbol{V}\| =: d_\mathrm{SU}(\boldsymbol{U}, \boldsymbol{V})$$
shall be N'_SU-computable (with $D(d_\mathrm{SU}) = \mathrm{SU}_\mu \times \mathrm{SU}_\mu$).
3) The *limit operator* $\lim_\mathrm{SU} : \ddot{\mathrm{SU}}_\mu \supseteq\!\to \mathrm{SU}_\mu$ defined through
$$\lim_\mathrm{SU}(\boldsymbol{U}_n)_{n\in\mathbb{N}} := \lim_{n\to\infty} \boldsymbol{U}_n$$
shall be N'_SU-computable (with $(\boldsymbol{U}_n)_{n\in\mathbb{N}} \in D(\lim_\mathrm{SU})$ if and only if the effective limit of $(\boldsymbol{U}_n)_{n\in\mathbb{N}}$ exists).
4) For all $1 \leq l, k \leq N$, $N \in \mathbb{N}$, the *basis-projection functional* $\beta_{l,k} : \mathrm{SU}_\mu(N) \to \mathbb{C}_\mu$ defined through
$$\beta_{l,k}(\boldsymbol{U}) := \langle \boldsymbol{e}_l | \boldsymbol{U} | \boldsymbol{e}_k \rangle$$
shall be N'_SU-computable (with $D(\beta_{l,k}) = \mathrm{SU}_\mu(N)$).

Then, given $N \in \mathbb{N}$, we say that N'_SU *faithfully implements* the *structure*

$$\big(\mathrm{SU}(N),\ \mathrm{gop}_\mathrm{SU},\ d_\mathrm{SU},\ (\beta_{l,k})_{1\leq l,k\leq N}\big) \tag{6}$$

if and only if $\mathsf{N}'_{\mathrm{SU}}$ satisfies Conditions 1, 2, 3, and 4.

The numbering N_{SU} (c.f. Definition 4) indeed faithfully implements the structure (6) for any $N \in \mathbb{N}$, as follows from the properties of $\mathbb{C}_\mu$ and $\mathsf{N}_{\mathbb{C}}$ (c.f., e.g., [22, Chapter 0, p. 12ff], [23, Section 4, p. 85ff], [24, Section 9.2.7, p. 312]). Further, note that all such numberings are *Turing equivalent*: Given any $\mathsf{N}'_{\mathrm{SU}} : \mathbb{N} \supseteq\!\!\rightarrow \mathrm{SU}_\mu$ that *faithfully implements* the *structure* (6), there exist μ-recursive functions $g_1 : D(\mathsf{N}_{\mathrm{SU}}) \to D(\mathsf{N}'_{\mathrm{SU}})$ and $g_2 : D(\mathsf{N}'_{\mathrm{SU}}) \to D(\mathsf{N}_{\mathrm{SU}})$ such that we have

$$\mathsf{N}_{\mathrm{SU}}(n) = [\mathsf{N}'_{\mathrm{SU}} g_1](n) \quad \text{and} \quad \mathsf{N}'_{\mathrm{SU}}(m) = [\mathsf{N}_{\mathrm{SU}} g_2](m)$$

for all $n \in D(\mathsf{N}_{\mathrm{SU}})$ and all $m \in D(\mathsf{N}'_{\mathrm{SU}})$. Accordingly, any statement about a function's Turing or Banach-Mazur computability with respect to *some* faithful numbering of SU_μ is equally valid for *any* faithful numbering of SU_μ. Finally, note that SU_μ is the *largest* subset of SU that exhibits a faithful numbering in the sense of Conditions 1, 2, 3, and 4 above.

Having established a computable-analysis framework for unitary matrices, we are now able to state a formalized version of the quantum compiler problem.

Quantum Compiler Problem (formalized) ——————

Given $N \in \mathbb{N}$, $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$, and any quantum compiler function $\mathtt{QC} : \mathrm{SU}_\mu \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$, find a μ-recursive function

$$g_{\mathtt{QC}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

such that for all $n \in D(\mathbb{N}_{\mathrm{SU}})$, $m \in \mathbb{N}$, we have $(n, m) \in D(g_{\mathtt{QC}})$, $g_{\mathtt{QC}}(n, m) \in D(\mathbb{N}_{\mathrm{SU}})$, and

$$\mathtt{QC}(\mathbb{N}_{\mathrm{SU}}(n), m) = [\mathbb{N}_{\mathrm{SU}} g_{\mathtt{QC}}](n, m).$$

In other words, find $g_{\mathtt{QC}}$ that realizes $\mathtt{QC}$ (with respect to $\mathbb{N}_{\mathrm{SU}}$) in the sense of Definition 1.

Below, we list several properties of the computable special (and, equivalently, general) unitary group. These properties will subsequently be relevant to our results and their proofs.

Summary 1 — Properties of SU_μ (and, equivalently, U_μ)**.**

§1. *The metric d_{SU} is Banach-Mazur computable and SU_μ is effectively closed with respect to d_{SU}.*

§2. *The group operation $\mathrm{gop}_{\mathrm{SU}}$ is iteratively computable: For every computable sequence $(\boldsymbol{V}_n)_{n\in\mathbb{N}} \subset \mathrm{SU}_\mu$, there exists a computable sequence $(\boldsymbol{U}_n)_{n\in\mathbb{N}} \subset \mathrm{SU}_\mu$ and μ-recursive functions $g : \mathbb{N} \to \mathbb{N}$, $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, such that for all $n, m, l \in \mathbb{N}$, we have*

$$\boldsymbol{U}_{g(n)} = \boldsymbol{V}_n \quad \text{and} \quad \boldsymbol{U}_{h(m,l)} = \boldsymbol{U}_m \boldsymbol{U}_l.$$

Accordingly, $\mathrm{gop}_{\mathrm{SU}}$ is also Banach-Mazur computable.

§3. *For $N \in \mathbb{N}$, $N \geq 2$, consider $(\boldsymbol{U}_m)_{m\in\mathbb{N}} \subset \mathrm{SU}_\mu(N)$ and assume there exist computable sequences $(\boldsymbol{V}_m)_{m\in\mathbb{N}} \subset \mathrm{SU}_\mu(N)$, $(\theta_{1,m})_{m\in\mathbb{N}}, \ldots, (\theta_{N,m})_{m\in\mathbb{N}} \subset \mathbb{R}_\mu$, such that we have*

$$\boldsymbol{U}_m = \sum_{n=1}^{N} \exp(\jmath 2\pi\theta_{n,m}) \boldsymbol{V}_m |\boldsymbol{e}_n\rangle\langle \boldsymbol{e}_n| \boldsymbol{V}_m^\dagger \text{ and } \ldots$$
$$\ldots \theta_{1,m} + \ldots + \theta_{N,m} = 0$$

for all $m \in \mathbb{N}$, where $|\boldsymbol{e}_1\rangle, \ldots, |\boldsymbol{e}_N\rangle \in \mathbb{C}_\mu^N$ is the computational basis (the requirement of $(\theta_{1,m})_{m\in\mathbb{N}}, \ldots, (\theta_{N,m})_{m\in\mathbb{N}}$ satisfying $\theta_{1,m} + \ldots + \theta_{N,m} = 0$ for all $m \in \mathbb{N}$ does not apply to $\mathrm{U}_\mu(N)$).

Then, $(\boldsymbol{U}_m)_{m\in\mathbb{N}}$ is a computable sequence of unitary matrices. In contrast, consider $\boldsymbol{U} \in \mathrm{SU}_\mu(N)$. There exist $\boldsymbol{V} \in \mathrm{SU}_\mu(N), \theta_1, \ldots, \theta_N \in \mathbb{R}_\mu$, have

$$\boldsymbol{U} = \sum_{n=1}^{N} \exp(\jmath 2\pi\theta_n) \boldsymbol{V} |\boldsymbol{e}_n\rangle\langle \boldsymbol{e}_n| \boldsymbol{V}^\dagger \text{ and } \theta_1 + \ldots + \theta_N = 0$$

(the consequence of $\theta_1, \ldots, \theta_N$ satisfying $\theta_1 + \ldots + \theta_N = 0$ may not apply to $\mathrm{U}_\mu(N)$).

§4. *Consider $M \in \mathbb{N} \setminus \{0\}$ and let $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,M}$ be any gate family. The function $\mathrm{bx}_M : \mathbb{N} \times \mathbb{N} \to [M]$ is μ-recursive. Hence, as follows from **§2**, defining*

$$(\boldsymbol{U}_m)_{m\in\mathbb{N}} : \boldsymbol{U}_m = \prod_{\infty}^{n=0} \underline{\boldsymbol{u}}\big(\mathrm{bx}_M(m, n)\big)$$

through primitive recursion [21, Definition 2.1 (V), p. 9] provides a computable sequence of unitary matrices. In other words, the monoid $\mathrm{G}(\underline{\boldsymbol{u}})$ is recursively enumerable.

To the best of the authors' knowledge, all quantum gates relevant in practice – including (but not limited to) all gates listen in [19, *Nomenclature and Notation*, p. xxixff], all gates whose matrix entries are algebraic numbers, and all gates whose Hamiltonian's matrix entries are algebraic numbers – belong to SU_μ. Assume we have $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$, $N \in \mathbb{N}$, i.e., all of the gates $\underline{\boldsymbol{u}}(1), \ldots, \underline{\boldsymbol{u}}(N)$ belong to SU_μ. Then, we must also have $\mathrm{G}(\underline{\boldsymbol{u}}) \subset \mathrm{SU}_\mu$. In other words, every quantum circuit we can build from any family of computable quantum gates corresponds to a computable unitary matrix itself. Furthermore, assume $\boldsymbol{U}_* \in \mathrm{SU}$ is the effective limit of some computable sequence $(\boldsymbol{U}_n)_{n\in\mathbb{N}} \subset \mathrm{G}(\underline{\boldsymbol{u}})$. Then, $\boldsymbol{U}_*$ must be an element of SU_μ in turn. Accordingly, all unitary transformations we can implement *effectively* – that is, such that we can control the approximation error by means of some algorithm – on any quantum computer that uses computable quantum gates are computable themselves. As stated just above, virtually all quantum gates relevant in practice belong to SU_μ. Thus, considering the computable unitary group (and the structure (6)) is not only a question of mathematical axioms, but also dictated by the practice of contemporary quantum engineering.

IV. Infeasibility of the Quantum Compiler Problem

Having introduced the computable unitary group, we are now equipped to analyze its relevant properties and, subsequently, investigate the quantum compiler problem on a formal basis. Proposition 1 and Theorem 1 below will characterize the computable unitary group's topological features in the context of Banach-Mazur computable functions. Corollary 1 and Corollary 2 will then address the quantum compiler problem and a special variant thereof concerning the *simulation of quantum systems* [19, Section 4.7, p. 204ff], respectively.

In the following, we denote the *unit interval* by $\mathcal{I}$, the *rational unit interval* by $\mathcal{I}_{\mathbb{Q}}$, and the *computable unit interval* by $\mathcal{I}_\mu$. That is, $\mathcal{I} := \{x \in \mathbb{R} : 0 \leq x \leq 1\}$, $\mathcal{I}_{\mathbb{Q}} := \{x \in \mathbb{Q} : 0 \leq x \leq 1\}$, and $\mathcal{I}_\mu := \{x \in \mathbb{R}_\mu : 0 \leq x \leq 1\}$.

Proposition 1: For $N \in \mathbb{N}$, let $\boldsymbol{H}_0, \boldsymbol{H}_1 \in \mathbb{C}_\mu^{N\times N}$ be any two trace-free skew-Hermitian matrices and define

$$H(\kappa, \tau, l) := \big(\exp(\kappa/l \boldsymbol{H}_0) \exp(\tau/l \boldsymbol{H}_1)\big)^l,$$

$$H_*(\kappa,\tau) := \lim_{l\to\infty} H(\kappa,\tau,l) = \exp(\kappa \boldsymbol{H}_0 + \tau \boldsymbol{H}_1),$$

for all $(\kappa,\tau) \in \mathcal{I}_\mu^2$, $l \in \mathbb{N}$. Then, we have the following.

1) The function $H : \mathcal{I}_\mu^2 \times \mathbb{N} \to \mathrm{SU}_\mu,\ (\kappa,\tau,l) \mapsto H(\kappa,\tau,l)$ is Banach-Mazur computable.
2) The function $H_* : \mathcal{I}_\mu^2 \to \mathrm{SU}_\mu,\ (\kappa,\tau) \mapsto H_*(\kappa,\tau)$ is Banach-Mazur computable. Specifically, given any two computable sequences $(\kappa_n)_{n\in\mathbb{N}}, (\tau_m)_{m\in\mathbb{N}} \subset \mathcal{I}_\mu$ and

$$(\boldsymbol{U}_{n,m,l})_{n,m,l\in\mathbb{N}} : \boldsymbol{U}_{n,m,l} = H(\kappa_n,\tau_m,l), \tag{7}$$

$$(\boldsymbol{U}_{n,m,*})_{n,m\in\mathbb{N}} : \boldsymbol{U}_{n,m,*} = H_*(\kappa_n,\tau_m), \tag{8}$$

the sequence $(\boldsymbol{U}_{n,m,l})_{n,m,l\in\mathbb{N}}$ converges effectively towards $(\boldsymbol{U}_{n,m,*})_{n,m\in\mathbb{N}}$ for $l \to \infty$.

Theorem 1: For $N \in \mathbb{N}$, consider a gate family $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$ and let $F : \mathcal{I}_\mu \to \mathrm{SU}_\mu$ be any Banach-Mazur computable function. Then, exactly one of the following claims holds true:

1) There exist $\boldsymbol{U}_* \in \mathrm{SU}_\mu \setminus \mathrm{G}(\underline{\boldsymbol{u}})$ and $x_* \in \mathcal{I}_\mu$ such that we have $\boldsymbol{U}_* = F(x_*)$.
2) There exists $\boldsymbol{U} \in \mathrm{G}(\underline{\boldsymbol{u}})$ such that for all $x \in \mathcal{I}_\mu$, we have $\boldsymbol{U} = F(x)$.

From a topological perspective, Theorem 1 does not come as a surprise. Unlike $\mathrm{G}(\underline{\boldsymbol{u}})$, the unit interval and the unitary group are *connected* sets. Thus, any function $F : \mathcal{I} \to \mathrm{G}(\underline{\boldsymbol{u}})$ must be either *discontinuous* or *constant*. If we instead consider a function $F : \mathcal{I} \to \mathrm{SU}$ that is *neither* discontinuous nor constant, its image cannot be a subset of (or equal to) $\mathrm{G}(\underline{\boldsymbol{u}})$. Since Banach-Mazur computability requires *effective continuity*, one may suspect similar principles to apply to Banach-Mazur computable functions $F : \mathcal{I}_\mu \to \mathrm{SU}_\mu$. The theory of *computable metric and Polish spaces* provides a comprehensive framework for analyzing mathematical problems of this kind – see, e.g., [24, Chapter 2, p. 29ff]. To keep the required preliminaries at a minimum, our proof of Theorem 1 (c.f. Appendix B) instead employs an explicit construction. For additional details on computable metric spaces, Banach-Mazur computable functions, and effective continuity, we alsorefer to [23], [30], [29] and [24, Chapter 9, p. 305ff].

Consider $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$, $N \in \mathbb{N}$, once more. Our proof of Theorem 1 directly provides an additional conclusion: For every computable sequence $(\boldsymbol{U}_n)_{n\in\mathbb{N}} \subset \mathrm{SU}_\mu$, there exists $\boldsymbol{U} \in \mathrm{SU}_\mu$ such that we have $\boldsymbol{U} \notin \{\boldsymbol{U}_n\}_{n\in\mathbb{N}}$. Accordingly, while the monoid $\mathrm{G}(\underline{\boldsymbol{u}})$ is recursively enumerable, the computable unitary group SU_μ is *not*. In this regard, the computable unitary group not being recursively enumerable is analogous to the unitary group being *uncountably infinite*.

Finally, note that adjusting Theorem 1 and its Corollaries 1 and 2 to the case of *infinite gate families* – that is, gate families $\underline{\boldsymbol{u}} : \mathbb{N} \to \mathrm{SU}_\mu$ that satisfy $\underline{\boldsymbol{u}}(0) = \mathrm{Id}$ and such that the sequence

$$(\boldsymbol{U}_n)_{n\in\mathbb{N}} : \boldsymbol{U}_n = \underline{\boldsymbol{u}}(n)$$

is a computable sequence of unitary matrices – is straightforward. Let $F : \mathcal{I}_\mu \to \mathrm{SU}_\mu$ be any *nonconstant* Banach-Mazur computable function and $(\boldsymbol{V}_n)_{n\in\mathbb{N}} \subset \mathrm{SU}_\mu$ *any* computable sequence. Then, our proof of Theorem 1 yields the following:

- There exists $x \in \mathcal{I}_\mu$ satisfying $F(x) \in \mathrm{SU}_\mu \setminus \{\boldsymbol{V}_n\}_{n\in\mathbb{N}}$.

Much analogous to the finite case, the infinite gate family $\underline{\boldsymbol{u}}$ generates a monoid $\mathrm{G}(\underline{\boldsymbol{u}})$. As follows from Summary 4, §2, $\mathrm{G}(\underline{\boldsymbol{u}})$ remains recursively enumerable: There exists a computable sequence $(\boldsymbol{V}_n)_{n\in\mathbb{N}} \subset \mathrm{SU}_\mu$ such that we have $\{\boldsymbol{V}_n\}_{n\in\mathbb{N}} = \mathrm{G}(\underline{\boldsymbol{u}})$. The adjusted variants of Theorem 1 and Corollaries 1 and 2 then follow.

A. Quantum Compiling and the Solovay-Kitaev Theorem

For $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$, $N \in \mathbb{N}$, consider a quantum compiler function $\mathtt{QC} : \mathrm{SU}_\mu \times \mathbb{N} \to \mathrm{G}(\underline{\boldsymbol{u}})$. Per definition, $\mathtt{QC}$ is a function that maps from $\mathrm{SU}_\mu \times \mathbb{N}$ to $\mathrm{G}(\underline{\boldsymbol{u}})$. Likewise, we may consider functions $\mathtt{SY} : \mathrm{SU}_\mu \times \mathbb{N} \times \mathbb{N} \to \{0,\dots,N\}$ such that for all $\boldsymbol{U} \in \mathrm{SU}$, $M \in \mathbb{N}$, and some μ-recursive function $\nu : \mathbb{N} \to \mathbb{N}$, we have

$$\left\| \boldsymbol{U} - \prod_{\nu(M)}^{l=0} \underline{\boldsymbol{u}}\big(\mathtt{SY}(\boldsymbol{U},M,l)\big) \right\| < \frac{1}{2^M}$$

(from Section II, recall that the inverted placement of "$l=0$" and "$\nu(M)$" indicates a *right-to-left* product). In other words, $\mathtt{SY}$ "synthesizes" a symbolic quantum circuit rather than computing the corresponding matrix representation directly. From an engineering perspective, the function $\mathtt{SY}$ is likely the more convenient option over $\mathtt{QC}$: For $\boldsymbol{U} \in \mathrm{SU}$, $M, L \in \mathbb{N}$ such that $L = \nu(M)$, the symbolic quantum circuit

$$\big(\mathtt{SY}(\boldsymbol{U},M,0),\dots,\mathtt{SY}(\boldsymbol{U},M,L)\big) \in \{0,\dots,N\}^{L+1}$$

serves as the machine-level instruction sequence required to physically "execute" the corresponding unitary transformation

$$\underline{\boldsymbol{u}}\big(\mathtt{SY}(\boldsymbol{U},M,L)\big) \cdots \underline{\boldsymbol{u}}\big(\mathtt{SY}(\boldsymbol{U},M,0)\big) \in \mathrm{G}(\underline{\boldsymbol{u}})$$

(c.f. Figure 1). It is nevertheless customary to define quantum compiler functions that map from $\mathrm{SU}_\mu \times \mathbb{N}$ to $\mathrm{G}(\underline{\boldsymbol{u}})$. In such cases, the distinction between symbolic quantum circuits and matrix representations is implicit: Given some input, the compiler function's definition usually allows for extracting a suitable symbolic quantum circuit from the individual steps required to evaluate the function. Arguably, this is why the definitions of established quantum compiler functions are often incorrectly labeled constructive.

Given $\underline{\boldsymbol{u}}$, $\mathtt{QC}$, and $\mathtt{SY}$ as above, consider $M, l \in \mathbb{N}$ arbitrary but fixed. For $\boldsymbol{U} \in \mathrm{SU}_\mu$, we may either define

$$F(\boldsymbol{U}) := \mathtt{QC}(\boldsymbol{U},M) \quad \text{or} \quad F(\boldsymbol{U}) := \mathtt{SY}(\boldsymbol{U},M,l),$$

which provides a function $F : \mathrm{SU}_\mu \to \mathrm{G}(\underline{\boldsymbol{u}})$ or $F : \mathrm{SU}_\mu \to \mathbb{N}$, respectively. In light of Corollary 1 below, we must consider the quantum compiler problem *unsolvable* for $\mathtt{QC}$ and $\mathtt{SY}$.

Corollary 1: For $N \in \mathbb{N}$, consider any gate family $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$.

1) Let $F : \mathrm{SU}_\mu \to \mathrm{G}(\underline{\boldsymbol{u}})$ be arbitrary. Then, we either have $F(\boldsymbol{U}) = F(\boldsymbol{V})$ for all $\boldsymbol{U}, \boldsymbol{V} \in \mathrm{SU}_\mu$ or F is not a Banach-Mazur computable function.
2) Let $F : \mathrm{SU}_\mu \to \mathbb{N}$ be arbitrary. Then, we either have $F(\boldsymbol{U}) = F(\boldsymbol{V})$ for all $\boldsymbol{U}, \boldsymbol{V} \in \mathrm{SU}_\mu$ or F is not a Banach-Mazur computable function.

For $N \in \mathbb{N}$, consider any universal gate family $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$ and observe the following. There *does* always exist a μ-recursive function $f_{\mathtt{QC}} : N^3 \supseteq\to \{0,\dots,N\}$ that satisfies

$$\left\| \mathsf{N}_{\mathrm{SU}}(n) - \prod_{\infty}^{l=0} \underline{\boldsymbol{u}}\big(f_{\mathtt{QC}}(n,m,l)\big) \right\| < \frac{1}{2^m}$$

for all $n \in D(\mathsf{N}_{\mathrm{SU}})$, $m \in \mathbb{N}$. At first glance, this might seem like an apparent contradiction to Corollary 1. However, $f_{\mathtt{QC}}$ does *not* define *any* function of the form $F : \mathrm{SU}_\mu \to \mathrm{G}(\underline{\boldsymbol{u}})$. For $m \in \mathbb{N}$ and $n_1, n_2 \in D(\mathsf{N}_{\mathrm{SU}})$ that satsify $\mathsf{N}_{\mathrm{SU}}(n_1) = \mathsf{N}_{\mathrm{SU}}(n_2)$, we will generally have

$$\prod_{\infty}^{l=0} \underline{\boldsymbol{u}}\big(f_{\mathtt{QC}}(n_1, m, l)\big) \neq \prod_{\infty}^{l=0} \underline{\boldsymbol{u}}\big(f_{\mathtt{QC}}(n_2, m, l)\big).$$

In fact, Corollary 1 implies the existence of an infinite number of such pairs (n_1, n_2). In other words, $f_{\mathtt{QC}}$ provides a gate sequence that depends on the exact description $n \in D(\mathsf{N}_{\mathrm{SU}})$ rather than (only) the represented matrix $\mathsf{N}_{\mathrm{SU}}(n) \in \mathrm{SU}_\mu$. On the level of functions from SU_μ to $\mathrm{G}(\underline{\boldsymbol{u}})$, it is thus not possible to make any statements about $f_{\mathtt{QC}}$ – concerning, e.g., it's computational complexity –, let alone optimize it in any form.

The impossibility of devising a constructive proof of the existence of any quantum compiler function is a direct consequence of our results. If such a proof were feasible, it would implicitly define an algorithm that computes the quantum compiler function in question (for a comprehensive treatment of the relation between algorithms and constructive proofs, see [25]). In the specific case of *C. M. Dawson's* and *M. A. Nielsen's* approach, computing the base-circuit approximation $\mathtt{SK}_0$ is a non-constructive operation. The role of $\mathtt{SK}_0$ in defining $\mathtt{QC}_{\mathtt{SK}}$ is crucial in this context: It provides the *terminal value* for the function's underlying *recursion*. Structurally, the terminal value is analogous to the *initial value* of an *iteration.* In turn, it is well established for diverse iterative numeric methods that calculating initial values is the pivotal breaking point concerning the method's computability [31]. In other words, the algorithmic feasibility of these methods relies on a priori knowledge of suitable input-dependent starting values for the iteration. Notably, however, an iterative method commonly requires one initial value per input. In contrast, multiple queries – for each possible input, the exact number grows polynomially in the desired accuracy – to the terminal-value function $\mathtt{SK}_0$ are necessary for evaluating $\mathtt{QC}_{\mathtt{SK}}$.

B. Simulating Quantum Systems and Adiabatic Evolutions

Observe that Theorem 1 equally applies even more directly to the case of simulating quantum systems with time-independent Hamiltonians and *adiabatic* evolutions. Both are often considered core applications of quantum computers, especially in areas such as quantum chemistry. For trace-free skew-Hermitian matrices $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$, we consider $\boldsymbol{H}_t := t\boldsymbol{H}_0 + (1-t)\boldsymbol{H}_1$, $t \in \mathcal{I}$. If the transition from $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$ is sufficiently slow, we have

$$|\boldsymbol{\phi}_t\rangle \approx \exp\big(\boldsymbol{H}_t\big)|\boldsymbol{\phi}_0\rangle, \quad 0 \leq t \leq 1, \tag{9}$$

c.f. (1) and (2). We define the *adiabatic arc*

$$\mathrm{AdA}(\boldsymbol{H}_0, \boldsymbol{H}_1) := \big\{\, \exp(\boldsymbol{H}_t) : t \in \mathcal{I}_\mu \big\} \subset \mathrm{SU}_\mu.$$

Accordingly, one may ask whether for any $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$, a "restricted" quantum compiler problem is solvable. In light of the subsequent Corollary 2, we must answer this question to the negative.

Corollary 2: For $N \in \mathbb{N}$, consider any gate family $\underline{\boldsymbol{u}} \in \mathrm{SU}_{\mu,N}$ and any trace-free skew-Hermitian matrices $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$. We have the following.

1) Let $F : \mathrm{AdA}(\boldsymbol{H}_0, \boldsymbol{H}_1) \to \mathrm{G}(\underline{\boldsymbol{u}})$ be arbitrary. Then, we either have $F(\boldsymbol{U}) = F(\boldsymbol{V})$ for all $\boldsymbol{U}, \boldsymbol{V} \in \mathrm{AdA}(\boldsymbol{H}_0, \boldsymbol{H}_1)$ or F is not a Banach-Mazur computable function.
2) Let $F : \mathrm{AdA}(\boldsymbol{H}_0, \boldsymbol{H}_1) \to \mathbb{N}$ be arbitrary. Then, we either have $F(\boldsymbol{U}) = F(\boldsymbol{V})$ for all $\boldsymbol{U}, \boldsymbol{V} \in \mathrm{AdA}(\boldsymbol{H}_0, \boldsymbol{H}_1)$ or F is not a Banach-Mazur computable function.

V. Conclusion

According to Corollary 1, quantum compiler functions in the sense of [8], [9], [10] *cannot* be implemented on Turing machines. This is due to the algebraic structure underlying the theory of gate-based quantum computing. Consequently, the analytic complexity analysis of the Solovay-Kitaev theorem is invalid on the level of Turing machines.

In the following, we provide a brief informal discussion on the relevant implications for "heuristic" attempts to implement quantum compilers based on approximate numerics. As a conclusion of our theory, the subsequent (qualitative) effects will necessarily be observable in any such implementation.

Output Instability. As discussed in Section IV, computability is strongly related to topological continuity. In essence, Corollary 1 is a result of the inherent discontinuity present in quantum compiler functions. In approximate numerics, this phenomenon manifests in output instabilities. That is, small disturbances in the input matrix may lead to significant variations in the computed gate sequence.

Pathologic Inputs. The previously discussed output instabilities also manifest in the context of computational complexity (as the analytic complexity analysis of the Solovay-Kitaev theorem does not hold true for Turing machines) and accuracy. For every heuristic quantum compiler, there exist well-behaved inputs $\boldsymbol{U}$ and ill-behaved inputs $\boldsymbol{V}$ with arbitrary small distance $\|\boldsymbol{U} - \boldsymbol{V}\|$, where the ill-behaved input leads to long computation times, long gate sequences, or poor accuracy. There is no a-priori way to distinguish whether an input is well- and ill-behaved.

Violation of Analytic Properties. Any analytic property that distinguishes a specific quantum compiler function is, for some inputs, violated in approximate implementations. For example, we may consider a quantum compiler function that minimizes the approximating gate-circuit's word length. For any approximate implementation of such a quantum compiler function, we can find inputs such that either the computed gate-circuit's word length is not minimal or the computed gate-circuit violates the accuracy constraint.

In principle, Turing machines allow for computing sequences of gate circuits that approximate a given computable unitary matrix up to any desired precision. However, such a computation does not abide the analytic model of quantum compiler functions as commonly found in the literature. An essential difference is that we cannot neglect the specific *description* of the unitary matrix to be approximated. Particularly, the complexity of this modified form of quantum compiling is an entirely open problem. However, the theory established within this paper is, in

principle, the suitable framework for approaching this problem in a mathematically rigorous manner.

REFERENCES

[1] Y. N. Böck, H. Boche, Z. G. del Toro, and F. H. P. Fitzek, "Feynman meets Turing: The infeasibility of digital compilers for gate-based quantum computing," in *Proc. IEEE Int. Conf. Commun. (ICC),* 2024, pp. 2440–2445.

[2] A. Montanaro, "Quantum algorithms: An overview," *Npj Quantum Inf.*, vol. 2, no. 1, pp. 1–8, 2016.

[3] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.,* 1994, pp. 124–134.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[5] P. W. Shor, "Why haven't more quantum algorithms been found?" *J. ACM (JACM)*, vol. 50, no. 1, pp. 87–90, 2003.

[6] P. W. Shor, *Progress in Quantum Algorithms.* Boston, MA, USA: Springer US, 2005, pp. 5–13.

[7] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Trans. Comput.*, vol. 70, no. 9, pp. 1363–1373, Sep. 2021.

[8] A. Y. Kitaev, "Quantum computations: Algorithms and error correction," *Russian Math. Surv.*, vol. 52, no. 6, p. 1191, Dec. 1997.

[9] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Inf. Comput.*, vol. 6, no. 1, pp. 81–95, Jan. 2006.

[10] A. Y. Kitaev, A. Shen, and M. N. Vyalyi, *Classical and Quantum Computation.* American Mathematical Society, Providence, Rhode Island, 2002.

[11] S. Bravyi, T. J. Yoder, and D. Maslov, "Efficient ancilla-free reversible and quantum circuits for the hidden weighted bit function," *IEEE Trans. Comput.*, vol. 71, no. 5, pp. 1170–1180, May 2022.

[12] V. Kliuchnikov, D. Maslov, and M. Mosca, "Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 161–172, Jan. 2016.

[13] E. Munoz-Coreas and H. Thapliyal, "Quantum circuit design of a t-count optimized integer multiplier," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 729–739, May 2019.

[14] R. Van Meter, K. Nemoto, and W. Munro, "Communication links for distributed quantum computation," *IEEE Trans. Comput.*, vol. 56, no. 12, pp. 1643–1653, Dec. 2007.

[15] M. Ying and Y. Feng, "An algebraic language for distributed quantum computing," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 728–743, Jun. 2009.

[16] H. J. Garcia and I. L. Markov, "Simulation of quantum circuits via stabilizer frames," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2323–2336, Aug. 2015.

[17] E. El-Araby et al., "Towards complete and scalable emulation of quantum algorithms on high-performance reconfigurable computers," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2350–2364, Aug. 2023.

[18] H. Boche, Y. N. Böck, Z. G. del Toro, and F. H. P. Fitzek, "Feynman meets Turing: The uncomputability of quantum gate-circuit emulation and concatenation," *IEEE Trans. Comput.*, vol. 74, no. 3, pp. 1053–1065, Mar. 2025.

[19] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge, U.K.: Cambridge Univ. Press, 2010.

[20] A. W. Harrow, B. Recht, and I. L. Chuang, "Efficient discrete approximations of quantum gates," *J. Math. Phys.*, vol. 43, no. 9, pp. 4445–4451, Sep. 2002.

[21] R. I. Soare, *Recursively Enumerable Sets and Degrees* (Perspectives in Mathematical Logic). Berlin, Germany: Springer-Verlag, 1987.

[22] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics* (Perspectives in Logic). Cambridge, U.K.: Cambridge Univ. Press, 1989.

[23] K. Weihrauch, *Computable Analysis* (Texts in Theoretical Computer Science: An EATCS Series). Springer-Verlag Berlin Heidelberg, 2000.

[24] V. Brattka and P. Hertling, *Handbook of Computability and Complexity in Analysis.* New York, NY, USA: Springer-Verlag, 2021.

[25] A. Bauer, "The realizability approach to computable analysis and topology," Ph.D. thesis, School of Computer Sci., Carnegie Mellon Univ., Pittsburgh, USA, 2000.

[26] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," in *Proc. London Math. Soc.,* Nov. 1936, vol. s2-42, no. 1, pp. 230–265.

[27] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem. A correction," in *Proc. London Math. Soc.,* vol. s2-43, no. 1, pp. 544–546, Jan. 1937.

[28] A. M. Turing, "Computability and λ-definability," *J. Symbolic Logic*, vol. 2, no. 4, pp. 153–163, Dec. 1937.

[29] P. Hertling, "A Banach–Mazur computable but not Markov computable function on the computable real numbers," *Ann. Pure Appl. Logic*, vol. 132, nos. 2–3, pp. 227–246, 2005.

[30] P. Hertling, "Banach-Mazur computable functions on metric spaces," in *Proc. Int. Workshop Computability Complexity Anal.*, New York, NY, USA: Springer-Verlag, 2000, pp. 69–81.

[31] G. H. Boche and A. Fono, "Turing meets Moore-Penrose: Computing the pseudoinverse on Turing machines," in *Proc. 35th Int. Workshop Operator Theory Appl. (IWOTA)*, Univ. of Kent. Springer, Aug. 2025.

[32] M. Suzuki, "Decomposition formulas of exponential operators and lie exponentials with some applications to quantum mechanics and statistical physics," *J. Math. Phys.*, vol. 26, no. 4, pp. 601–612, 1985.

Yannik N. Böck (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degree in electronics engineering and information technology from the Technical University of Munich, Munich, Germany, in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree with the Chair of Theoretical Information Technology with Technical University of Munich (TUM). Since 2019, he has been a member of the Research and Teaching Staff of the Chair of Theoretical Information Technology, TUM. His research interests include quantum information theory and the applications of computability theory in engineering.

Holger Boche (Fellow, IEEE) received the Dipl.-Ing. degree in electrical engineering, Dipl. Math. degree in mathematics, and the Dr.-Ing. degree in electrical engineering from the Technische Universität Dresden, Dresden, Germany, in 1990, 1992, and 1994, respectively. In 1998, he received the Dr. rer. nat. degree in pure mathematics from the Technische Universität Berlin, Berlin, Germany. He is currently a Full Professor with the Chair of Theoretical Information Technology, Technische Universität München, Munich, Germany, which he joined in 2010. Since 2021, he and Frank H.P. Fitzek have jointly headed the BMBF research hub 6G-life. He was elected a member of the German Academy of Sciences (Leopoldina) in 2008 and the Berlin Brandenburg Academy of Sciences and Humanities in 2009. He received the "Innovation Award" from the Vodafone Foundation in 2006 and the Gottfried Wilhelm Leibniz Prize from the German Research Foundation in 2008.

Zoe Garcia del Toro received the M.Sc. degree in quantum science and technology from the Technical University of Munich and Ludwig-Maximilian University of Munich, Germany, in 2024. She is currently working toward the Ph.D. degree with LIP6, CNRS, Sorbonne Université, Paris, France, where she works on higher-order quantum operations and quantum information theory. During her studies in Munich, she worked as a Student Research Assistant with the Chair of Theoretical Information Technology, focusing on applications of computability theory. Her research interests include intersection of quantum information, higher-order quantum transformations, and the mathematical foundations of quantum theory.

Frank H.P. Fitzek (Fellow, IEEE), received the diploma in electrical engineering from RWTH Aachen, Germany, in 1997. He received the Ph.D. degree in electrical engineering from the Technical University Berlin, in 2002. In 2002, he embarked on his professorial journey with the University of Ferrara, Italy, and further expanded his academic influence to Aalborg University in 2003 as a Professor. He is a Professor and Chair of the "Deutsche Telekom Chair of Communication Networks" with TU Dresden, Leads with the forefront of telecommunications research in Germany. As the spokesperson for the DFG Cluster of Excellence CeTI and the 6G-life hub, his contributions have significantly shaped the field of communication networks. Educationally, his research interests include 5G/6G communication networks, in-network computing, network coding, compressed sensing, post-Shannon theory, quantum and molecular communication, and immersive human-machine interaction in virtual environments.