Feynman Meets Turing: The Uncomputability of Quantum Gate-Circuit Emulation and Concatenation

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

***Abstract*—We investigate the feasibility of computing quantum gate-circuit emulation (QGCE) and quantum gate-circuit concatenation (QGCC) on digital hardware. QGCE serves the purpose of rewriting gate circuits comprised of gates from a varying input gate set to gate circuits formed of gates from a fixed target gate set. Analogously, QGCC serves the purpose of finding an approximation to the concatenation of two arbitrary elements of a varying list of input gate circuits in terms of another element from the same list. Problems of this kind occur regularly in quantum computing and are often assumed an easy task for the digital computers controlling the quantum hardware. Arguably, this belief is due to analogical reasoning: The classical Boolean equivalents of QGCE and QGCC are natively computable on digital hardware. In the present paper, we present two insights in this regard: Upon applying a rigorous theory of computability, QGCE and QGCC turn out to be uncomputable on digital hardware. The results remain valid when we restrict the set of feasible inputs for the relevant functions to one parameter families of fixed gate sets. Our results underline the possibility that several ideas from quantum-computing theory may require a rethinking to become feasible for practical implementation.**

***Index Terms*—Quantum circuits, quantum-circuit emulation, quantum-circuit concatenation, turing machines, effective analysis.**

Received 21 February 2024; revised 23 September 2024; accepted 12 November 2024. Date of publication 27 November 2024; date of current version 12 February 2025. The work of Holger Boche and Yannik N. Böck was supported in part by 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 additionally supported by the BMBF Quantum Programm QD-CamNetz under Grant 16KISQ077, QuaPhySI under Grant 16KIS1598K, and QUIET under Grant 16KISQ093. The work of Zoe Garcia del Toro was supported by German Research Foundation (DFG) within the Gottfried Wilhelm Leibniz Prize Programme under Grant BO 1734/20-1 and in part by DFG Projects under Grant BO 1734/35-1 and Grant STA 864/9-1. The work of Frank H. P. Fitzek was supported in part by German Research Foundation (DFG, Deutsche Forschungs-gemeinschaft) as part of Germany's Excellence Strategy – EXC2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden. The authors also acknowledge the financial support 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 identification number 16KIS1998K; project Q-TREX, project identification number 16KISR027; and project QUIET, project identification number 16KISQ092. An earlier version of this paper was presented in part at IEEE International Symposium on Information Theory (ISIT), 2024 [DOI: 10.1109/ISIT57864.2024.10619233]. Recommended for acceptance by N. Revol. *(Corresponding author: Yannik N. Böck.)*

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

Yannik N. Böck and Zoe Garcia del Toro are with the Chair of Theoretical Information Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: yannik.boeck@tum.de; zoe.garcia@tum.de).

Frank H. P. Fitzek is with the Deutsche Telekom Chair of Communication Networks, BMBF Research Hub 6G-life, the Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop (CeTI)," and the 5G Lab Germany, Technische Universität Dresden, D-01062 Dresden, Germany (e-mail: frank.fitzek@tu-dresden.de).

Digital Object Identifier 10.1109/TC.2024.3506861

I. Introduction

QUANTUM computing is often considered one of the most promising recent technological advances. According to theoretical findings, quantum computers may be able to accomplish some computing tasks faster than traditional digital information processing. Prominent potential instances of such computing tasks include prime factorization of large integers in cryptography, or the simulation of protein behavior in computational chemistry. The scientific community anticipates that quantum computing will dramatically boost global processing capacity for a variety of fields and applications, such as economics optimization and traffic routing.

Despite the fact that quantum computers fundamentally aim to exploit non-classical physical phenomena, *universal quantum computers* (as opposed to quantum annealers, for example, which are largely application specific) conceptually share most of their architectural principles with classical computers. In the gate-model of classical computing, programs process information by applying Boolean functions to bit-strings. In turn, the hardware implements these functions through circuits of logic gates. Choosing a *universal* set of basic gates ensures *functional completeness*, i.e., the existence of a corresponding circuit of logic gates for every possible boolean function. Modern programming languages include commands for modifying the computer's memory down to the level of individual bits. In engineering practice however, the larger part programming happens on a higher level of abstraction, with the relevant bit-level logic being created automatically by the compiler. This form of automatization is possible due to the discrete nature of Boolean functions, which makes it feasible to algorithmically rewrite different circuit representations of each function into each other. The compiler can optimize the

concatenation – the formal analogon of successively applying functions – of Boolean functions to require a smaller number of basic gates, with the program's input-output behavior being retained. Particularly, the principle of functional completeness ensures that different universal sets of logic gates are able to *emulate* each other.

In *gate-based quantum computing*, information is processed by successively applying quantum gates – which, in their specific order of application, form a *quantum gate-circuit* – to the computer's quantum memory. The process of applying these gates is controlled by an algorithm that runs on a digital computer, a principle mirrored in emerging quantum programming languages [2]. Commonly, these languages incorporate the novel feature of manipulating qubits into traditional programming paradigms. Each available quantum gate corresponds to a unitary operator on some finite-dimensional Hilbert space, with the entirety of all such operators forming the quantum equivalent of classical Boolean functions. Following the seminal publications on the efficient approximability of quantum algorithms in the gate-circuit model [3], [4], contemporary research largely focuses on basic sets of quantum gates with regards to different performance metrics, refining theoretical models for representing quantum algorithms through gate-circuits, and mitigating computational errors introduced by inaccuracies in the quantum gates, especially with regards to scalability. Optimized gate-circuit representations for specific quantum algorithms were e.g. considered in [5], [6], [7]. Simulating quantum-algorithms on classical hardware constitutes a widespread method in the development of quantum-computing platforms. Since the amount of required computational resources is usually large, techniques for optimizing such simulations have been an active field of study as well [8], [9]. Advanced and interdisciplinary topics in gate-based quantum computing concern, for example, distributed information processing architectures [10], [11]. However, the design of gate-circuit implementations of quantum algorithms remains challenging, despite the comprehensive research efforts [2].

In contrast to the set of Boolean functions, Hilbert spaces contain a continuum. Accordingly, there exists an uncountable variety of possibly universal gate sets, each of which may exhibit vastly different properties regarding efficiency, error tolerance, and overall practical feasibility. Further, as it is only possible to represent a countable number of unitary operators through finite gate circuits, it is necessary to implement general quantum algorithms – that is, arbitrary unitary operators – in an approximate manner. Since classical computing acts on discrete mathematical objects, it cannot natively account for the analytic principles of continuity and approximation. Accordingly, it is not immediately evident whether digital hardware is able to logically control the execution of operations such as quantum-circuit emulation or quantum-circuit concatenation in a mathematically correct manner. Despite this interdependence of classical and quantum information processing being present within all contemporary models of quantum computing, large parts of the relevant theoretical literature approaches quantum computing through abstract analytic mathematics. Classical computing, on the other hand, is entirely characterized by Turing's theory of computability, entailing the framework of effective analysis. Although ideas such as quantum Turing machines were discussed conceptually [12], rigorous mathematical treatments of quantum computing on the basis of effective analysis remain sparse. In [13], we have approached this issue and established a suitable framework that considers the task of compiling quantum algorithms to gate-circuits. In the present work, we aim to extend this framework to the problems of computing quantum gate-circuit emulation functions and quantum gate-circuit concatenation functions, which we formally introduce in the following section. We prove that in contrast to their classical domain, quantum gate-circuit emulation functions and quantum gate-circuit concatenation functions cannot be implemented by means of digital algorithms. For quantum gate-circuit emulation functions, this fundamental difference is schematically depicted in Fig. 1.

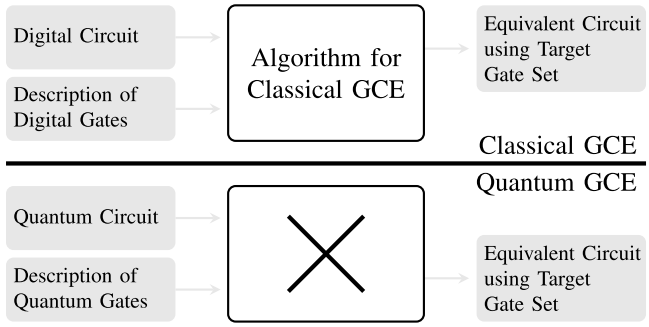

Fig. 1. Schematic comparison of classical gate-circut emulation and quantum gate-circuit emulation. In the gate-circuit emulation problem, (in both the classical and the quantum variant) the target gate set is fixed. The input of a gate-circuit emulation algorithm consists of a description of an arbitrary gate set, together with a circuit consisting only of gates within that set. The corresponding output consist of an equivalent circuit that consists only of gates within the target gate set. In contrast to the classical variant, there does not exist a (digital) algorithm that satisfies these requirements for the quantum variant of the problem.

The remainder of this paper is structured as follows. In Section II, we and motivate and formally introduce the problem of computing quantum gate-circuit emulation functions and quantum gate-circuit functions. Further, we precisely formalize the corresponding research questions with regards to our theory. In III and IV, we introduce the relevant theoretical background. The main contribution of this paper is presented Sections V and VI, where we state the Theorems answering the research questions posed in Section II. Finally, we conclude the paper by a brief subsumption of our results in Section VII.

A. General Remarks, Nomenclature, and Notation

Let $\mathcal{A}$ be an arbitrary set. By $\mathrm{P}\mathcal{A}$, we denote the *power set* of $\mathcal{A}$. That is, for an arbitrary set $\mathcal{A}'$, we have $\mathcal{A}' \in \mathrm{P}\mathcal{A}$ if and only if we also have $\mathcal{A}' \subseteq \mathcal{A}$.

Let $\mathcal{A}$ and $\mathcal{B}$ be two arbitrary sets. We refer to a mapping $F : \mathrm{D}(F) \to \mathcal{B}$ as *partial* (with regard to $\mathcal{A}$) for $\mathrm{D}(F) \subseteq \mathcal{A}$, and write $F : \mathcal{A} \supseteq\!\to \mathcal{B}$. If $\mathrm{D}(F) = \mathcal{A}$, we refer to F as a *total* and write $F : \mathcal{A} \to \mathcal{B}$. Since $\mathcal{A} \subseteq \mathcal{A}$ trivially holds true, every total

mapping is also partial (but not vice versa). Additionally, we refer to $\mathrm{D}(F)$ as the *domain of definition* – in short, *domain* – of F.

A complex number $z \in \mathbb{C}$ satisfies $z = \mathrm{Re}(z) + \jmath\,\mathrm{Im}(z)$, where $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$ are the *real* and *imaginary* part of z, respectively. Accordingly, $\jmath$ denotes the *imaginary unit*.

Throughout this paper, we consider arbitrary but fixed finite-dimensional Hilbert spaces over $\mathbb{C}$, w.l.o.g. defined through $\mathcal{H} = \mathrm{span}\{|1\rangle, \ldots, |N\rangle\},\ N = \dim \mathcal{H} \in \mathbb{N}$. The vectors $|1\rangle, \ldots, |N\rangle$ constitute a distinguished orthonormal basis of $\mathcal{H}$. Additionally, the vectors $|1\rangle, \ldots, |N\rangle$ provide an isomorphism between the set $\mathbb{C}^{N\times N}$ and the set of linear operators acting on $\mathcal{H}$. Thus, we shall implicitly consider any linear operator acting on $\mathcal{H}$ an element of $\mathbb{C}^{N\times N}$, i.e., a complex-valued matrix. If $U^\dagger = U^{-1}$ is satisfied by a linear operator $U : \mathcal{H} \to \mathcal{H}$, it is referred to as a *unitary operator*. Here, $U^\dagger$ and U^{-1} denote the adjoint and inverse of U, respectively. We denote the set of unitary operators acting on $\mathcal{H}$ by $\mathcal{U}$. Unless indicated otherwise, we consider the standard matrix norm

$$||\cdot|| : \mathbb{C}^{N\times N} \to \mathbb{R},\ ||A|| := \sup_{|\psi\rangle\in\mathcal{H}, |\psi\rangle\neq 0} \frac{\sqrt{\langle\psi| A^\dagger A |\psi\rangle}}{\sqrt{\langle\psi|\psi\rangle}}.$$

II. Problem Formulation

In the bit-string model of traditional computing, which is the classical analogue to quantum gate circuits, information processing is based on Boolean functions $B \in \mathcal{B}(N)$, where

$$\mathcal{B}(N) := \left\{B : \{0,1\}^N \to \{0,1\}^N\right\},\ N \in \mathbb{N}.$$

A classical gate set $\mathcal{G}_{\mathrm{cl}} := \{G_1, \ldots, G_M\},\ M \in \mathbb{N}$, is called universal – which, as indicated in the introduction, is equivalent to the functional completeness of $\mathcal{G}_{\mathrm{cl}}$ [14], [15] – if, for all $B \in \mathcal{B}(N)$, there exists a tuple $\left(G_{m(1)}, \ldots, G_{m(L)}\right)$ with $\boldsymbol{y} := (m(1), \ldots, m(L))$, $m \in \mathbb{N}$, such that for all $\boldsymbol{x} \in \{0,1\}^N$, we have

$$B(\boldsymbol{x}) = \left[\mathcal{G}_{\mathrm{cl}}(\boldsymbol{y})\right](\boldsymbol{x}) = \left[G_{m(1)} \cdots G_{m(L)}\right](\boldsymbol{x}),$$

where we write $\mathcal{G}_{\mathrm{cl}}(\boldsymbol{y}) := G_{m(1)} \cdots G_{m(L)}$ for the composition of $G_{m(1)}, \ldots, G_{m(L)}$, $m \in \mathbb{N}$. The bit-string model of digital computing has a longstanding history in engineering and has been extensively researched, following the seminal work by Claude Shannon [16].

For all N, the extension of the singleton gate sets $\mathcal{G}_{\mathrm{nand}} := \{\mathtt{NAND}\}$ and $\mathcal{G}_{\mathrm{nor}} := \{\mathtt{NOR}\}$ obtained through arbitrary input-output wiring – that is, gate sets consisting of all gates of the form

$$\boldsymbol{x} \mapsto \left(x_1, \ldots x_{n-1}, \mathtt{NAND}(x_m, x_l), x_{n+1}, \ldots, x_N\right),$$

$1 \leq n, m, l \leq N$, or sets consisting of all gates of the form

$$\boldsymbol{x} \mapsto \left(x_1, \ldots x_{n-1}, \mathtt{NOR}(x_m, x_l), x_{n+1}, \ldots, x_N\right),$$

$1 \leq n, m, l \leq N$, – are each functional complete with respect to $\mathcal{B}(N)$ (c.f. also Fig. 1). For simplicity, we will not distinguish between $\mathcal{G}_{\mathrm{nand}}$ and $\mathcal{G}_{\mathrm{nor}}$, respectively, and their extensions to N variables.

In digital logics, either of $\mathcal{G}_{\mathrm{nand}}$ and $\mathcal{G}_{\mathrm{nor}}$ are commonly used for practical implementations of integrated circuits. The following problems arise naturally and are crucial to classical computing:

1) From an arbitrary boolean function B, compute a corresponding circuit $\boldsymbol{y}$ using the gate set $\mathcal{G}_{\mathrm{nand}}$, i.e., a circuit $\boldsymbol{y}$ such that we have $\mathcal{G}_{\mathrm{nand}}(\boldsymbol{y}) = B$.
2) From an arbitrary boolean function B, compute a corresponding circuit $\boldsymbol{y}$ using the gate set $\mathcal{G}_{\mathrm{nor}}$, i.e., a circuit $\boldsymbol{y}$ such that we have $\mathcal{G}_{\mathrm{nor}}(\boldsymbol{y}) = B$.
3) From an arbitrary circuit $\boldsymbol{y}$ using the gate set $\mathcal{G}_{\mathrm{nand}}$, compute a corresponding circuit $\boldsymbol{y}'$ using the gate set $\mathcal{G}_{\mathrm{nor}}$ (i.e., a circuit $\boldsymbol{y}'$ such that we have $\mathcal{G}_{\mathrm{nand}}(\boldsymbol{y}) = \mathcal{G}_{\mathrm{nor}}(\boldsymbol{y}')$), or vice versa.

The same questions may equivalently be formulated for any other pair $(\mathcal{G}_{\mathrm{cl}}, \mathcal{G}'_{\mathrm{cl}})$ of universal gate sets.

We can solve all of the three abovementioned problems algorithmically. Particularly, since the set $\mathcal{B}(N)$ is finite for all $N \in \mathbb{N}$, we can do so in dependence of $(\mathcal{G}_{\mathrm{cl}}, \mathcal{G}'_{\mathrm{cl}})$. Denote the set of (general) tuples on natural numbers by $\mathbb{N}^*$. That is, with "()" denoting the empty tuple, we have

$$\mathbb{N}^* := \left\{()\right\} \cup \mathbb{N} \cup \mathbb{N}^2 \cup \mathbb{N}^3 \ldots = \bigcup\nolimits_{n\in\mathbb{N}} \mathbb{N}^n.$$

Then, for $M \in \mathbb{N}$ and a fixed universal gate set $\mathcal{G}_{\mathrm{cl}}$, we can find Turing-computable functions $\Phi_{\mathrm{cl},M} : \mathbb{N}^M \times \mathrm{P}\mathcal{B}(N) \supseteq\to \mathbb{N}^*$ such that for all pairs $(\boldsymbol{y}, \mathcal{G}'_{\mathrm{cl}})$ that satisfy $\mathcal{G}'_{\mathrm{cl}} \in \mathrm{P}\mathcal{B}(N)$ and $\boldsymbol{y} \in \{1, \ldots, |\mathcal{G}'_{\mathrm{cl}}|\}^M$, we have $\mathcal{G}'_{\mathrm{cl}}(\boldsymbol{y}) = \mathcal{G}_{\mathrm{cl}}(\Phi_{\mathrm{cl},M}(\boldsymbol{y}, \mathcal{G}'_{\mathrm{cl}}))$, i.e., $\Phi_{\mathrm{cl},M}$ emulates gate-circuits of length M from arbitrary gate sets $\mathcal{G}'_{\mathrm{cl}}$ by means of gate-circuits from the gate set $\mathcal{G}_{\mathrm{cl}}$.

The use of specific functionally complete gate sets is not only relevant from a practical point of view, where the employed gate sets are chosen such that a specific technology is easy to implement, but also from a theoretical one. Especially in the context of classical complexity theory, expressing boolean functions by means of different basic gates is essential to answering fundamental mathematical questions. In turn, complexity theory is highly relevant for fields such as modern cryptography. Often, we require to represent boolean functions in a specific way, such as the *conjugate normal form* (CNF). These are symbolic equations of the type

$$\boldsymbol{x} \mapsto \bigwedge_{m=1}^{M} \underbrace{\left(b_{m,1}(x_1) \vee \ldots \vee b_{m,N}(x_N)\right)}_{=:b_m(\boldsymbol{x})} \tag{1}$$

for some $M \in \mathbb{N}$ and a family $(b_{m,n})_{m,n=1}^{M,N}$ with values in the alphabet $\{\text{'}\mathtt{O}\text{'}, \text{'}\mathtt{I}\text{'}, \text{'}\mathtt{N}\text{'}\}$, where we define

$$b_{m,n}(x) := \begin{cases} 0, & \text{if } b_{m,n} = \mathtt{O}, \\ x, & \text{if } b_{m,n} = \mathtt{I}, \\ \neg x, & \text{if } b_{m,n} = \mathtt{N}. \end{cases} \tag{2}$$

The resulting boolean function is thus uniquely characterized through the family $(b_{m,n})_{m,n=1}^{M,N}$.

The general *SAT* decision problems, which consist of determining whether for a given function B, there exists $\boldsymbol{x} \in \{0,1\}^N$ such that we have $B(\boldsymbol{x}) = 1$, are a fundamental example of a class of complexity-theory problems with high relevance to many fields of computer science and information technology. According to a theorem by *Cook* and *Levin*, SAT is *NP complete*

[17], [18]. If the boolean function is given in CNF – i.e., in terms of a family $(b_{m,n})_{m,n=1}^{M,N}$ as above – even more can be said. For example, we may consider families $(b_{m,n})_{m,n=1}^{M,N}$ such that for all $1 \leq m \leq M$, no more than three of the symbols $b_{m,1}, \ldots, b_{m,N}$ are different from '0'. Intuitively, this means that each of the terms $b_m(\boldsymbol{x})$, $1 \leq m \leq M$, depends on no more than three variables. If SAT is restricted to functions of this form, the resulting decision problem *3SAT* remains NP complete. Likewise, we can further restrict the the SAT problem by considering families $(b_{m,n})_{m,n=1}^{M,N}$ such that each of the terms $b_m(\boldsymbol{x})$, $1 \leq m \leq M$, depends on no more than two variables. Then, the resulting restricted SAT problem *2SAT* is polynomial-time decidable, i.e., lies within the complexity class P. Computing CNF representations of boolean functions is thus a relevant problem in classical complexity theory.

The prove of SAT's NP completeness provides a constructive method to transform any non-deterministic polynomial-time Turing machine into a corresponding boolean function. The number of inputs the Turing machine *accepts* – that is, the number of inputs for which the Turing machine eventually halts – equals the number of values $\boldsymbol{x} \in \{0,1\}^N$ for which the corresponding boolean function equals one. This way, we can fully characterize the complexity class $\#P$ *(Sharp-P)* through the SAT problem. Further, #3SAT *(Sharp-3SAT)* and #2SAT *(Sharp-2SAT)* are both $\#P$ complete, adding another relevant facet to SAT. From a theoretical perspective, the correspondence between SAT and #SAT *(Sharp-SAT)* allows for a deeper understanding of the structural properties of the underlying set of accepted inputs.

For quantum computing, the problem of emulation is equally relevant. There exist multiple potential approaches to implement gate based quantum computing, each resulting in different universal sets of basic gates. Hardware manufacturers may use these degrees of freedom, and they do so in practice. If one manufacturer wants to benchmark their implementation versus those of other manufacturers, they must solve an emulation problem. Such comparisons are crucial for the assessment of different forms of quantum computing hardware. Further, as indicated in Section I, the programming of contemporary gate-based quantum computers employs specialized programming languages, which incorporate high-level operations – such as the quantum Fourier transform, or a quantum phase estimation, for example – that manipulate *logical* qubits. These operations and qubits are visible to the programmer and entirely virtual. Ultimately, a compiler must map the program written by the programmer to a sequence of physical operations implemented in the available hardware, resulting in the manipulation of *physical qubits*. In order to do so, it requires an interpretation of the high-level operations in terms of their action with respect to the chosen computational basis. Depending on the degree of variability in this interpretation, the compiler must solve a general or restricted gate-circuit emulation problem.

In gate-based quantum computing, a quantum algorithm is always characterized by a unitary matrix $U \in \mathcal{U}$ on some finite dimensional Hilbert space $\mathcal{H}$. The execution of any such algorithm is performed by sequentially applying a list of atomic operations. These atomic operations $U_1, \ldots, U_N \in \mathcal{U}$, $N \in \mathbb{N}$, which are themselves unitary matrices, are referred to as quantum gates and form the quantum computer's gate set $\mathcal{G}_\mathrm{q} := \{U_1, \ldots, U_N\}$. The resulting product $U_{n(1)}U_{n(2)}\cdots U_{n(M)}$, $M \in \mathbb{N}$, with $U_{n(m)} \in \mathcal{G}_\mathrm{q}$ for all $1 \leq m \leq M$, is referred to as gate-circuit. Since the set of unitary matrices on any Hilbert space is continuous, it cannot be fully represented by means of discrete structures such as gate-circuits. Thus, $\mathcal{G}_\mathrm{q}$ is usually chosen *universal*, i.e., such that the group generated by $\mathcal{G}_\mathrm{q}$ is dense in $\mathcal{U}$, see below. In this case, any $U \in \mathcal{U}$ can, in principle, be approximated up to any desired accuracy by applying a suitable sequence of quantum gates. In formal terms, if $\mathcal{G}_\mathrm{q}$ is universal, then, for all $U \in \mathcal{U}$ and all $\delta \in \mathbb{R}$ with $\delta > 0$, there exists a suitable gate-circuit $\mathcal{G}_\mathrm{q}(\boldsymbol{y}) := U_{n(1)}U_{n(2)}\cdots U_{n(M)}$ with $\boldsymbol{y} := (n(1), \ldots, n(M))$, $M \in \mathbb{N}$, such that we have

$$\left\|U - U_{n(1)}U_{n(2)}\cdots U_{n(M)}\right\| = \left\|U - \mathcal{G}_\mathrm{q}(\boldsymbol{y})\right\| < \delta,$$

Thus, in contrast to the case of classical gate circuits, the model of gate-based quantum-computing is inherently approximate in nature.

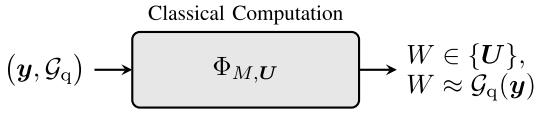

Fig. 2. Schematic depiction of (general) quantum gate-circut emulation as considered in Problem 1.

In the present work, we will extend the theory established in [13] to the problem of computing quantum gate-circuit emulation functions, which is the quantum-computational equivalent of the classical emulation problem described in the preceding paragraph. For tuples $\boldsymbol{U} = (U_1, \ldots, U_N) \subset \mathcal{U}$, we define $|\boldsymbol{U}| := N$ and $\{\boldsymbol{U}\} := \{U_1, \ldots, U_N\}$. With some abuse of notation, we write $U \in \boldsymbol{U}$ whenever we have $U \in \{\boldsymbol{U}\}$. Given a number $M \in \mathbb{N}$, consider a fixed δ-net $\boldsymbol{U} \in \mathcal{D}_\delta$ (c.f. Section V for the formal definition of $\mathcal{D}_\delta$), potentially generated by some gate set $\mathcal{G}_\mathrm{q}$. We consider partial functions $\Phi_{M,\boldsymbol{U}} : \mathbb{N}^M \times \mathrm{P}\mathcal{U} \supseteq\!\!\rightarrow \{\boldsymbol{U}\}$, such that for all pairs $(\boldsymbol{y}, \mathcal{G}_\mathrm{q})$ that satisfy $\mathcal{G}_\mathrm{q} \in \mathrm{P}\mathcal{U}$ and $\boldsymbol{y} \in \{1, \ldots, |\mathcal{G}_\mathrm{q}|\}^M$, we have $\|\mathcal{G}_\mathrm{q}(\boldsymbol{y}) - \Phi_{M,\boldsymbol{U}}(\boldsymbol{y}, \mathcal{G}_\mathrm{q})\| < \delta$. In particular, we ask if it is possible to implement any such function, which we subsequently refer to as *(general) quantum gate-circuit emulation* (QGCE) *functions*, in terms of an algorithm on digital hardware. Within the formal model of Turing computability, this translates to the following problem.

Problem 1 [c.f. Fig. 2]. *Given $M \in \mathbb{N}$ and a δ-net $\boldsymbol{U} \subset \mathcal{U}$, consider general QGCE functions $\Phi_{M,\boldsymbol{U}}$. Is any of these functions Turing computable?*

As discussed above, the classical gate-circuit emulation problem is solvable on Turing machines. It is not immediately evident whether this is the case for QGCE as well, since it is not generally possible to implement quantum gates from one (varying) universal gate set $\mathcal{G}'_\mathrm{q}$ through finite gate-circuits from another (fixed) universal gate set $\mathcal{G}_\mathrm{q}$.

As a demonstrative example, consider a quantum programming language $\mathcal{L} := \{\text{'X'}, \text{'Y'}, \text{'Z'}\}^*$, i.e., the individual programs consist of finite strings with letters from the alphabet $\{\text{'X'}, \text{'Y'}, \text{'Z'}\}$. For $\mathcal{H} = \text{span}\{|1\rangle, |2\rangle\}$ and $\theta \in \mathbb{R}$, denote the rotation gates in the Bloch basis by $R_{\mathrm{X}}(\theta)$, $R_{\mathrm{Y}}(\theta)$, and $R_{\mathrm{Z}}(\theta)$. That is, we have

$$R_{\mathrm{X}}(\theta) := \exp(-\imath/2\theta \mathrm{X}),\ R_{\mathrm{Y}}(\theta) := \exp(-\imath/2\theta \mathrm{Y}),$$
$$R_{\mathrm{Z}}(\theta) := \exp(-\imath/2\theta_{\mathrm{Z}}),$$

where X, Y and Z are the non-idenitity Pauli matrices. Then, an interpretation of the language $\mathcal{L}$ in the above sense may consist of the correspondences $\text{'X'} \mapsto R_{\mathrm{X}}(\theta)$, $\text{'Y'} \mapsto R_{\mathrm{Y}}(\theta)$, and $\text{'Z'} \mapsto R_{\mathrm{Z}}(\theta)$ for some parameter $\theta \in \mathbb{R}$. On the hardware level, we may be able to implement a finite list of finite sequences $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M\} \subseteq \{1, 2, 3\}^L$, $M, L \in \mathbb{N}$, of rotations in the Bloch basis for a finite number of parameters $\{\theta_1, \ldots, \theta_K\}$, $K \in \mathbb{N}$, such that the resulting list of physical operations forms a δ-net. Defining $\mathcal{G}_{\mathrm{Bl}}(\theta) := \{\mathcal{G}_{\mathrm{Bl}}(1|\theta), \mathcal{G}_{\mathrm{Bl}}(2|\theta), \mathcal{G}_{\mathrm{Bl}}(3|\theta)\}$ with

$$\mathcal{G}_{\mathrm{Bl}}(1|\theta) := R_{\mathrm{X}}(\theta),\ \mathcal{G}_{\mathrm{Bl}}(2|\theta) := R_{\mathrm{Y}}(\theta),$$
$$\mathcal{G}_{\mathrm{Bl}}(3|\theta) := R_{\mathrm{Z}}(\theta),$$

we have $\boldsymbol{U}_{\mathrm{Bl}} := \big(\mathcal{G}_{\mathrm{Bl}}(\boldsymbol{y}_m|\theta_k)\big)_{1\leq m\leq M, 1\leq k\leq K}$ for said list. On the programming level, however, the programmer might want to be able to choose θ from the continuous range of computable real numbers $\mathbb{R}_\mu$ (see Section IV for their formal definition). In this case, a compiler for the language $\mathcal{L}$ needs to be able to compute a suitable mapping $(\mathtt{L}, \theta) \mapsto U \in \boldsymbol{U}_{\mathrm{Bl}}$ for $\mathtt{L} \in \mathcal{L}$ and $\theta \in \mathbb{R}_\mu$, leading to a *restricted* version of Problem 1. We will formalize and investigate this restricted QGCE problem in Section V, proving its computational infeasibility. The relevant no-go statement (c.f. Theorem 2) concerns general one-parameter families of gate sets, thus applying to $\mathcal{G}_{\mathrm{Bl}}(\theta) : \theta \in \mathbb{R}_\mu$ as defined above. Accordingly, it is not possible to compute mappings of the form $(\mathtt{L}, \theta) \mapsto U \in \boldsymbol{U}_{\mathrm{Bl}}$ as above on digital hardware.

In the above context, the δ-net $\boldsymbol{U} \subset \mathcal{U}$ represents the aggregate of operations that we can execute on the quantum hardware we have available. This aggregate may consist of a list of gate-circuits, each of which is formed by the atomic operations from $\mathcal{G}_{\mathrm{q}}$. For example, $\mathcal{U}$ may simply contain all gate-circuits of depth $M \in \mathbb{N}$, i.e.,

$$\{\boldsymbol{U}\} = \Big\{\mathcal{G}_{\mathrm{q}}(\boldsymbol{y}) : \boldsymbol{y} \in \{1, \ldots, |\mathcal{G}_{\mathrm{q}}|\}^M\Big\}.$$

Since we may want to exclude specific gate-circuits from implementation by our hardware – for example, for reasons of efficiency – $\boldsymbol{U}$ may likewise consist of a specific subset of the set of all gate-circuits of depth M. In both cases, $\boldsymbol{U}$ is generally *not* closed under matrix multiplication, which mathematically corresponds to physically concatenating the relevant gate-circuits. Since the concatenation of gate-circuits is, just as in the realm of traditional hardware, fundamental to quantum computing, we require a solution to this obstacle.

For tuples $\boldsymbol{U} = (U_1, \ldots, U_N) \subset \mathcal{U}$ and $u \in \{1, \ldots, |U|\}$, we denote $\boldsymbol{U}(u) := U_u$ (this is consistent with the notation $\mathcal{G}_{\mathrm{q}}(\boldsymbol{y})$ for gate sets $\mathcal{G}_{\mathrm{q}}$ when considering u as a singleton tuple $\boldsymbol{y} = (u)$, in which case we have $\boldsymbol{U}(\boldsymbol{y}) = U_u$). Given $\delta > 0$, we consider partial functions $\Psi_\delta : \mathbb{N} \times \mathbb{N} \times \mathcal{D}_\delta \supseteq\!\!\rightarrow \mathcal{U}$ such that

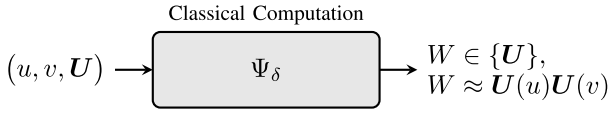

Fig. 3. Schematic depiction of (general) quantum gate-circuit concatenation as considered in Problem 2.

for all tuples $(u, v, \boldsymbol{U})$ that satisfy $\boldsymbol{U} \in \mathcal{D}_\delta$, $u \in \{1, \ldots, |\boldsymbol{U}|\}$, and $v \in \{1, \ldots, |\boldsymbol{U}|\}$, we have $\Psi_\delta(u, v, \boldsymbol{U}) \in \{\boldsymbol{U}\}$ as well as $\|\boldsymbol{U}(u)\boldsymbol{U}(v) - \Psi_\delta(u, v, \boldsymbol{U})\| < \delta$. Subsequently, we will refer to such functions as *(general) quantum gate-circuit concatenation* (QGCC) *functions*. As for QGCE functions, we ask if it is possible to implement any QGCC function in terms of an algorithm on digital hardware. Within the formal model of Turing computability, this translates to the following problem.

Problem 2 [c.f. Fig. 3]. *Given $\delta > 0$, consider general QGCC functions Ψ_δ. Is any of these functions Turing computable?*

While – as is the case for QGCE – the classical equivalent to QGCC is solvable on Turing machines, it is not evident whether the same holds true for QGCC itself.

In Sections V and VI, we establish a theory that provides a thorough characterization Problems 1 and 2 in the context of computable analysis, answering both problems to the negative. As indicated above, we will also consider *restricted-input* QGCE and QGCC functions, which, in contrast to their general counterparts, are defined in the context of one-parameter families of gate sets. The demonstrative example of the language L indicates the special relevance of such families. As a consequence of Schrödinger's equation, the majority of practically relevant quantum gates belong to such families, with θ corresponding to the *evolution time* of the associated physical system.

III. Preliminaries on Turing's Theory of Computability

As discussed in Section I, computable analysis is founded on Turing's theory of computability [19], [20]. The *Turing machine* is a model of an idealized digital computer. A computational operation cannot be practically implemented on a digital computer if it cannot be theoretically implemented on a Turing machine. Gate-based quantum computing architectures employ digital hardware to control the application of quantum gates. In contrast to unitary operators, which are elements of continuous mathematical spaces, digital hardware is intrinsically discrete. Accordingly, Turing machines do not natively understand the idea of a quantum algorithms. Instead, we require a precise formalism to encode a suitable subset of unitary operators into tuples of natural numbers.

The set of all partial mappings $g : \mathbb{N}^n \supseteq\!\!\rightarrow \mathbb{N}$, $n \in \mathbb{N}$, that a Turing machine can implement comprises the set of *μ-recursive functions* (c.f. [21, Definition 2.1, p. 8, Definition 2.2, p. 10] for the more common alternative definition, as well as [22] for their equivalence). In short, the set of μ-recursive functions is the smallest set of functions $g : \mathbb{N}^n \supseteq\!\!\rightarrow \mathbb{N}$, $n \in \mathbb{N}$, that contains

the *successor function*, all *constant functions*, and all *projection functions* and is closed with respect to *composition*, *primitive recursion*, and *unbounded search*.

Turing machines are state-based automata, i.e., they carry out computations in sequential steps. In this context, the domain $\mathrm{D}(g)$ of the μ-recursive function $g:\mathbb{N}^n \supseteq\!\!\rightarrow \mathbb{N}$, $n\in\mathbb{N}$, computed by some Turing machine has a distinguished interpretation. Precisely, it equals the set of those inputs for which the Turing machine halts its computation after a finite number of computing steps. As a result of the well-known *halting problem*, the *indicator function*

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

of $\mathrm{D}(g)$ is *not* necessarily a μ-recursive function itself.

Define the bijective mapping $\mathrm{K}:\mathbb{N}\times\mathbb{N}\to\mathbb{N}\backslash\{0\}$ according to $\mathrm{K}(n,m):=2^n(2m+1)$. Then, K is a (total) μ-recursive function, and the there exists a unique pair (κ_1,κ_2) of (total) μ-recursive functions such that $n=\kappa_1(\mathrm{K}(n,m))$ and $m=\kappa_2(\mathrm{K}(n,m))$ hold true for all $n,m\in\mathbb{N}$. Further, define $[\![\cdot]\!]:\mathbb{N}^*\to\mathbb{N}$ according to $[\![()]\!]:=0$ and

$$[\![\boldsymbol{x}]\!]:=\mathrm{K}\big(m_1,\cdots,\mathrm{K}\big(m_{n-1},\mathrm{K}\big(m_n,0\big)\big)\cdots\big)$$

for all $\boldsymbol{x}:=(m_1,\ldots,m_n)\in\mathbb{N}^*,\ n\in\mathbb{N}$. Then, $[\![\cdot]\!]:\mathbb{N}^*\to\mathbb{N}$ is bijective (and total). A family $(\Gamma m)_{m\in\mathbb{N}}$ of functions $\Gamma m:\mathbb{N}^*\to\mathbb{N}^*$, $m\in\mathbb{N}$ is called effective enumeration of μ-recursive functions if there exists a μ-recursive function $h:\mathbb{N}\supseteq\!\!\rightarrow\mathbb{N}$ such that for every tuple $(g_1,\ldots,g_L)$ of μ-recursive functions $g_1,\ldots,g_l:\mathbb{N}^n\supseteq\!\!\rightarrow\mathbb{N}$, $n,L\in\mathbb{N}$, there exists $m_0\in\mathbb{N}$ such that

$$\mathrm{D}\big(\Gamma m_0\big)\cap\mathbb{N}^l=\bigcap_{l=1}^{m}\mathrm{D}(g_l)=\Big\{\boldsymbol{x}\in\mathbb{N}^n:[\![m_0,\boldsymbol{x}]\!]\in\mathrm{D}(h)\Big\}$$

(where we write $[\![m_0,\boldsymbol{x}]\!]$ for $[\![m_0,m_1,\ldots,m_n]\!]$ with some abuse of notation) is satisfied and, for all $\boldsymbol{x}\in\mathrm{D}(\Gamma m_0)\cap\mathbb{N}^l$, we have $\Gamma m_0(\boldsymbol{x})=[\![h([\![m_0,\boldsymbol{x}]\!])]\!]^{-1}=(g_n(\boldsymbol{x}),\ldots,g_l(\boldsymbol{x}))$. We call m_0 a *program* for $(g_1,\ldots,g_L)$. Up to a μ-recursive permutation of $\mathbb{N}$, the effective enumeration $(\Gamma m)_{m\in\mathbb{N}}$ is unique. Thus, we fix $(\Gamma m)_{m\in\mathbb{N}}$ for the remainder of the article.

IV. Preliminaries on Effective Analysis and Computable Unitary Operators

Applying Turing's theory to the domain of real and complex numbers, we arrive at the discipline of *computable analysis*. For a comprehensive introduction, we refer to [23], [24].

A sequence $(r_n)_{n\in\mathbb{N}}\subset\mathbb{Q}$ is called a *computable sequence of rational numbers* if there exist (total) μ-recursive functions $g,h_1,h_2:\mathbb{N}\to\mathbb{N}$ such that

$$r_n=\frac{(-1)^{g(n)}\cdot h_1(n)}{h_2(n)+1}$$

is satisfied for all $n\in\mathbb{N}$. For $m\in\mathbb{N}$, an *m-fold computable sequence of rational numbers* is defined through (total) μ-recursive functions $g,h_1,h_2:\mathbb{N}^m\to\mathbb{N}$ in m arguments.

A complex number z is called *computable* if there exists a pair $((r_n^{\mathrm{re}})_{n\in\mathbb{N}},(r_n^{\mathrm{im}})_{n\in\mathbb{N}})$ of computable sequences of rational numbers and a (total) μ-recursive function $\xi\colon\mathbb{N}\to\mathbb{N}$ such that $|z-(r_n^{\mathrm{re}}+\jmath r_n^{\mathrm{im}})|<2^{-N}$ holds true for all $n,N\in\mathbb{N}$ that satisfy $n\geq\xi(N)$. We denote the set of such numbers by $\mathbb{C}_\mu$. A number $z\in\mathbb{C}_\mu$ with $\mathrm{Im}(z)=0$ is called a computable real number, the set of which we denote by $\mathbb{R}_\mu$. Finally, a sequence $(z)_{n\in\mathbb{N}}\subset\mathbb{C}$ is called *computable sequence of complex numbers* if there exists a pair $((r_{n,m}^{\mathrm{re}})_{n,m\in\mathbb{N}},(r_{n,m}^{\mathrm{im}})_{n,m\in\mathbb{N}})$ of computable double sequences of rational numbers and a (total) μ-recursive function $\xi:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ such that $|z-(r_{n,m}^{\mathrm{re}}+\jmath r_{n,m}^{\mathrm{im}})|<2^{-M}$ holds true for all $n,m,M\in\mathbb{N}$ that satisfy $m\geq\xi(M)$.

As indicated in the beginning of this section, Turing machines do not have native access to mathematical objects that belong to continuous spaces in a nontrivial manner, such as complex numbers with irrational real and imaginary part. In order to establish a notion of computability for such spaces, we need to specify a natural-number based encoding of a suitable subset of their elements. Given some analytically well-defined set $\mathcal{A}$ as well as some subset $\mathcal{A}_\mu\subseteq\mathcal{A}$, we refer to a partial surjective mapping $[\cdot]_\mu^{\mathcal{A}}:\mathbb{N}\supseteq\!\!\rightarrow\mathcal{A}_\mu$ as *numbering* for the set $\mathcal{A}_\mu$. Recalling the above definitions, observe that any computable real or complex numbers is ultimately characterized by a tuple of μ-recursive functions. The effective enumeration $(\Gamma m)_{m\in\mathbb{N}}$ (c.f. Section III) thus provides numberings $[\cdot]_\mu^{\mathbb{C}}:\mathbb{N}\supseteq\!\!\rightarrow\mathbb{C}_\mu$ and $[\cdot]_\mu^{\mathbb{R}}:\mathbb{N}\supseteq\!\!\rightarrow\mathbb{R}_\mu$.

More generally, *Machine-readable languages* and *canonical numberings* [25], which are strongly related to the principles of *computability structures* [23, Chapter 2.1, p. 80ff], (traditional) *numberings* [24, Chapter 1.4, p. 12], and *naming systems* [24, Definition 2.3.1, p. 33], provide a thorough formalization of numberings for arbitrary analytically well-defined sets. In modern *type theory*, a pair consisting of an arbitrary set and an associated numbering is referred to as *modest set*, see e.g. [26]. Given a list $\mathcal{A}_{\mu,1},\ldots,\mathcal{A}_{\mu,N},\mathcal{A}_{\mu,N+1}$, $N\in\mathbb{N}$, of arbitrary sets with associated numberings, a partial function

$$F:\mathcal{A}_{\mu,1}\times\ldots\times\mathcal{A}_{\mu,N}\supseteq\!\!\rightarrow\mathcal{A}_{\mu,N+1}$$

is called Turing computable if there exists a μ-recursive function $g:\mathbb{N}^N\supseteq\!\!\rightarrow\mathbb{N}$ such that we have

$$F\left([m_1]_{\mu,1}^{\mathcal{A}},\ldots,[m_N]_{\mu,N}^{\mathcal{A}}\right)=[g(\boldsymbol{x})]_{\mu,N+1}^{\mathcal{A}}\tag{3}$$

for all $\boldsymbol{x}=(m_1,\ldots,m_N)\in\mathbb{N}^N$, provided the left-hand side of (3) is well-defined.

It remains to provide a numbering for suitable subset of $\mathcal{U}$. In [13], we established a numbering of a set $\mathcal{U}_\mu\subset\mathcal{U}$, the elements of which we referred to as *computable unitary operators*. This numbering is natural in the sense that it adheres to the analytic structure of $\mathcal{U}$ and, of any such numberings, numbers the largest possible set $\mathcal{U}_\mu\subset\mathcal{U}$. The method is analogous to *M. B. Pour-El* and *J. I. Richards'* approach of computability structures [23, Chapter 2.1, p. 80ff] for Banach-spaces (c.f. Section III), which can be considered a cornerstone of computable analysis.

Definition 1: Let $(U_n)_{n\in\mathbb{N}}\subset\mathcal{U}$ be a sequence of unitary operators that is dense in $\mathcal{U}$. If, for all $m,l\in\{1,\ldots,\dim\mathcal{H}\}$, the sequence $(z_{m,l,n})_{n\in\mathcal{N}}$, defined through $z_{m,l,n}:=\langle m|U_n|l\rangle$

for all $n \in \mathbb{N}$, is a computable sequence of complex numbers, we say that $(U_n)_{n\in\mathbb{N}}$ induces a computability structure on $\mathcal{U}$:

1) An operator $U \in \mathcal{U}$ is called computable if there exist (total) μ-recursive functions $g, \xi : \mathbb{N} \to \mathbb{N}$ such that we have $\|U - U_{g(n)}\| < 2^{-N}$ for all $n, N \in \mathbb{N}$ with $n \geq \xi(N)$, and we denote the set of such operators by $\mathcal{U}_\mu$;
2) For Γ (c.f. Section III) fixed, we define the numbering $[\cdot]^{\mathcal{U}}_\mu : \mathbb{N} \supseteq\!\to \mathcal{U}_\mu,\ m \mapsto [m]^{\mathcal{U}}_\mu$ such that $[m]^{\mathcal{U}}_\mu \mathcal{U} := U$ iff we have $\Gamma m = (g, \xi)$ for a pair (g, ξ) that determines U in the sense of Point 1.

For $U \in \mathcal{U}$, a function $\lambda \mapsto U^\lambda \in \mathcal{U},\ \lambda \in \mathbb{R}$ is *exponential* if it satisfies $U^0 = \mathrm{Id}$, $U^1 = U$, and $U^s U^t = U^{s+t}$ for all $s, t \in \mathbb{R}$. Note that all such functions are of the form $\lambda \mapsto \exp(-\imath/\hbar 2\pi\lambda \mathrm{H})$, where H is a hermitian operator on $\mathcal{H}$. Given $U \in \mathcal{U}$, corresponding operators H are not unique. For each $U \in \mathcal{U}$, there exist multiple exponential functions, each of which is characterized by a hermitian operator H.

Lemma 1: Let $(U_n)_{n\in\mathbb{N}} \subset \mathcal{U}$ and $(V_n)_{n\in\mathbb{N}} \subset \mathcal{U}$ be any two sequences that induce computability structures on $\mathcal{U}$. Further, denote the corresponding sets of computable unitary operators by $\mathcal{U}_\mu \subset \mathcal{U}$ and $\mathcal{V}_\mu \subset \mathcal{U}$, and the corresponding numberings by $[\cdot]^{\mathcal{U}}_\mu$ and $[\cdot]^{\mathcal{V}}_\mu$, respectively. The following holds true.

1) We have $\mathcal{U}_\mu = \mathcal{V}_\mu$, i.e., the set $\mathcal{U}_\mu$ does not depend on the specific choice of $(U_n)_{n\in\mathbb{N}}$.
2) There exists a recursive function $g : \mathbb{N} \supseteq\!\to \mathbb{N}$ such that whenever $[m]^{\mathcal{U}_\mu}$ is well-defined for some $m \in \mathbb{N}$, we have $[m]^{\mathcal{U}}_\mu = [g(m)]^{\mathcal{V}}_\mu$. Since $(U_n)_{n\in\mathbb{N}}$ and $(V_n)_{n\in\mathbb{N}}$ were chosen arbitrarily, $[\cdot]^{\mathcal{U}}_\mu$ and $[\cdot]^{\mathcal{V}}_\mu$ are effectively equivalent.
3) For all $n, m \in \{1, \ldots, \dim \mathcal{H}\}$, the corresponding basis projection $U \mapsto \langle n|U|m\rangle \in \mathbb{C}_\mu$ is Turing computable with domain $\mathcal{U}_\mu$. The group operation $(U, V) \mapsto UV \in \mathcal{U}_\mu$ is Turing computable with domain $\mathcal{U}_\mu \times \mathcal{U}_\mu$. For all $U \in \mathcal{U}_\mu$, each exponential function $\lambda \mapsto U^\lambda \in \mathcal{U}_\mu$ is Turing computable with domain $\mathbb{R}_\mu$.

As a consequence of Lemma 1, any statement about Turing computability with respect to a specific computability structure on $\mathcal{U}_\mu$ is equally valid for *all* computability structures on $\mathcal{U}$. Thus, we can ignore the specific choice of $(U_n)_{n\in\mathbb{N}}$ in our analysis. Furthermore, $\mathcal{U}_\mu$ includes virtually all quantum gates that are relevant in practice. For details on Definition 1 and Lemma 1, we refer the reader to [13].

We conclude the preliminaries on computability theory and computable analysis by stating two technical lemmas we will subsequently require. A function $F : [0, 1] \cap \mathbb{R}_\mu \supseteq\!\to \{0, 1\}$ is called Turing computable if there exists a μ-recursive function $g_F : \mathbb{N} \supseteq\!\to \{0, 1\}$ such that for all $n \in \mathbb{N}$ within the relevant domain, we have $F([n]^{\mathbb{R}}_\mu) = g(n)$.

The following lemma can be found in an analogous form in [27, Lemma 1, p. 2260]. It comprises the key ingredient in proving the uncomputability of *weak quantum gate-circuit functions* (not to be confused with QGCE and QGCC functions as introduced in the present article) as considered in [13, Lemma 1], we will require it subsequently to prove Theorems 1, 2, 3, and 4.

Lemma 2: Let $F : [0, 1] \cap \mathbb{R}_\mu \to \{0, 1\}$ be a (total) function such that $F(0) \neq F(1)$ holds true. Then, F is not Turing computable function.

As an indirect consequence of Lemma 2, any discontinuous Turing-computable function $F : [0, 1] \cap \mathbb{R}_\mu \supseteq\!\to \mathbb{R}_\mu$ must be undefined at any point of discontinuity. The following lemma is readily obtained from [23, Proposition 0, p. 14].

Lemma 3: Let x be any computable real number. Then, the partial function

$$\mathbb{1}_{<x} : \mathbb{R}_\mu \supseteq\!\to \{0, 1\},\ \lambda \mapsto \mathbb{1}_{<x}(\lambda) := \begin{cases} 1, & \text{if } \lambda < x, \\ 0, & \text{if } \lambda > x, \end{cases}$$

with domain $\mathrm{D}(\mathbb{1}_{<x}) = \mathbb{R}_\mu \setminus \{x\}$ is a Turing-computable function. □

V. The Uncomputability of Quantum Gate-Circuit Emulation Functions

By $\mathcal{U}^*_\mu$, we denote the set of tuples of computable unitary operators. With "()" denoting the empty tuple, we have

$$\mathcal{U}^*_\mu := \{()\} \cup \mathcal{U}_\mu \cup \mathcal{U}^2_\mu \cup \mathcal{U}^3_\mu \ldots = \bigcup\nolimits_{n\in\mathbb{N}} \mathcal{U}^n_\mu.$$

Next, define $[\cdot]^{\mathcal{U}}_\star : \mathbb{N} \supseteq\!\to \mathcal{U}^*_\mu,\ m \mapsto [m]^{\mathcal{U}}_\star$ such that for $m \in \mathbb{N}$, $[m]^{\mathcal{U}}_\star$ is well-defined with $[m]^{\mathcal{U}}_\star := (U_1, \ldots, U_N)$ iff we have $(U_1, \ldots, U_N) = ([m_1]^{\mathcal{U}}_\mu, \ldots, [m_N]^{\mathcal{U}}_\mu)$ for $(m_1, \ldots, m_N) = [\![m]\!]^{-1}$. For $\delta > 0$, we have $\boldsymbol{U} \in \mathcal{D}_\delta$ iff we have $\delta > \min\{\|U - \boldsymbol{U}(n)\| : n \in \{1, \ldots, |\boldsymbol{U}|\}\}$ for all $U \in \mathcal{U}$.

In the following, we consider general QGCE functions as addressed in Problem 1. That is, given $M \in \mathbb{N}$ and $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}^*_\mu$, partial functions $\Phi_{M,\boldsymbol{U}} : \mathbb{N}^M \times \mathrm{P}\mathcal{U}_\mu \supseteq\!\to \{\boldsymbol{U}\}$, such that for all pairs $(\boldsymbol{y}, \mathcal{G}_\mathrm{q})$ that satisfy $\mathcal{G}_\mathrm{q} \in \mathrm{P}\mathcal{U}_\mu$ and $\boldsymbol{y} \in \{1, \ldots, |\mathcal{G}_\mathrm{q}|\}^M$, we have $\|\mathcal{G}_\mathrm{q}(\boldsymbol{y}) - \Phi_{M,\boldsymbol{U}}(\boldsymbol{y}, \mathcal{G}_\mathrm{q})\| < \delta$. Observe that $\Phi_{M,\boldsymbol{U}}$ is defined for arbitrary arguments $\mathcal{G}_\mathrm{q} \subset \mathrm{P}\mathcal{U}_\mu$. Hence, we can reduce the problem of *quantum compiling* (c.f. [13]) to the function $\Phi_{M,\boldsymbol{U}}$. Accordingly, we obtain the following.

Theorem 1 (General QGCE Functions): Given $M \in \mathbb{N}$ and $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}^*_\mu$, let $\Phi_{M,\boldsymbol{U}} : \mathbb{N}^M \times \mathrm{P}\mathcal{U}_\mu \supseteq\!\to \{\boldsymbol{U}\}$ be a general QGCE function. Then, $\Phi_{M,\boldsymbol{U}}$ is not Turing computable.

As indicated in Section II, the classical equivalent to QGCE is solvable algorithmically. In fact, this remains true even if the circuits feasible for input are not upper bounded in depth. Given a universal classical gate set $\mathcal{G}_\mathrm{cl}$, we can always find a Turing-computable function

$$\Phi_{\mathrm{cl},*} : \mathbb{N}^* \times \mathrm{P}\mathcal{B}(N) \supseteq\!\to \mathbb{N}^*$$

such that for all pairs $(\boldsymbol{y}, \mathcal{G}'_\mathrm{cl})$ that satisfy $\mathcal{G}'_\mathrm{cl} \in \mathrm{P}\mathcal{B}(N)$ and $\boldsymbol{y} \in \{1, \ldots, |\mathcal{G}'_\mathrm{cl}|\}^{M(\boldsymbol{y})}$ for any $M(\boldsymbol{y}) \in \mathbb{N}$, we have

$$\mathcal{G}'_\mathrm{cl}(\boldsymbol{y}) = \mathcal{G}_\mathrm{cl}(\Phi_{\mathrm{cl},*}(\boldsymbol{y}, \mathcal{G}'_\mathrm{cl})).$$

The circuit specified through $\Phi_{\mathrm{cl},*}(\boldsymbol{y}, \mathcal{G}'_\mathrm{cl})$ emulates the circuit $\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})$ by means of gates from the set $\mathcal{G}_\mathrm{cl}$.

Since the set of bit-strings of length $N \in \mathbb{N}$ is finite, an algorithm that implements $\Phi_{\mathrm{cl},*}$ is conceptually simple. In principle, – that is, leaving considerations regarding computational complexity aside – it is sufficient to generate a truth-table for $\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})$ by computing $[\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})](\boldsymbol{x})$ for all $\boldsymbol{x} \in \{0, 1\}^{\{N\}}$. Because $\mathcal{G}'_\mathrm{cl}$ is part of the algorithms input, we have knowledge of

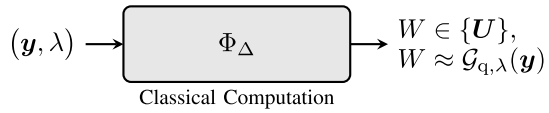

Fig. 4. Schematic depiction of restricted-input quantum gate-circut emulation functions with $\Delta = (\mathcal{G}_\mathrm{q}, V, M, \boldsymbol{U})$, as considered in Theorem 2.

the behavior of each of the gates that form the circuit $\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})$. Thus, evaluating $[\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})](\boldsymbol{x})$ for arbitrary $\boldsymbol{x} \in \{0,1\}^{\{N\}}$ is indeed feasible. As indicated in Section II the set of Boolean functions in N variables is is finite as well. Hence, provided $\mathcal{G}_\mathrm{cl}$ is universal, there exists $L \in \mathbb{N}$ such that

$$\mathcal{B}(N) = \left\{\mathcal{G}_\mathrm{cl}(\boldsymbol{y}') : \boldsymbol{y}' \in \{1, \ldots, |\mathcal{G}_\mathrm{cl}|\}^L\right\}$$

i.e., using gates from $\mathcal{B}(N)$, we can synthesize every Boolean function in N variables through a circuit of depth less than or equal to m. Thus, successively computing the truth-tables of $\mathcal{G}_\mathrm{cl}(\boldsymbol{y}')$ for all $\boldsymbol{y}' \in \{1, \ldots, |\mathcal{G}_\mathrm{cl}|\}^L$, we will eventually find a circuit whose truth-table coincides with the one of $\mathcal{G}'_\mathrm{cl}(\boldsymbol{y})$.

In practice, enumerating truth-tables is infeasible for Boolean functions of large depth and width due to the required computational resources. Nevertheless, enumeration argument proves the classical emulation problem's algorithmic solvability in principle. Returning to the realm of quantum computing, we find that by Theorem 1, the (general) QGCE problem is algorithmically intractable even for circuits of fixed width $N = 2$. Notably, this is not for reasons that concern complexity, but due to fundamental uncomputability. Particularly, as follows from the proof of Lemma 2 (which, in turn, will be crucial in deriving Theorem 1), implementing a (general) QGCE function is at least as hard as solving the halting problem (c.f. [13] for details). Note that it is possible to compute a sequence $\mathcal{G}_\mathrm{q}(\cdot|2), \mathcal{G}_\mathrm{q}(\cdot|3), \ldots, \mathcal{G}_\mathrm{q}(\cdot|N), \ldots$ consisting of universal gate sets $\mathcal{G}_\mathrm{q}(\cdot|N) \in \mathrm{P}\mathcal{B}(N)$ for quantum gate circuits of arbitrary width $N \in \mathbb{N}$. The uncomputability of general QGCE functions is a consequence of the continuous nature of the set $\mathcal{U}$, which makes the construction of computable unitary operators necessary in the first place. In contrast, Boolean functions are natively accessible to digital hardware.

The proof of Theorem 1 will mirror the continuous nature of unitary operators as follows. Given the input $(\boldsymbol{y}, \mathcal{G}_\mathrm{q})$ of a (general) QGCE function, the actual circuit description $\boldsymbol{y}$ remains constant, while the $\mathcal{G}_\mathrm{q}$ varies along a continuous curve. Accordingly, the relevant method allows for extracting a stricter version of Theorem 1 that fixes the input up to one parameter $\lambda \in \mathbb{R}_\mu$. Given a possibly empty gate set $\mathcal{G}_\mathrm{q} = \{U_1, \ldots, U_N\}$, $N \in \mathbb{N}$, and any non-identity $V \in \mathcal{U}_\mu$, we consider parameterized gate sets $\mathcal{G}_{\mathrm{q},\lambda} := \{U_1, \ldots, U_N, V^\lambda\}$, $\lambda \in \mathbb{R}_\mu$. For $M \in \mathbb{N}$ and any $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}_\mu^*$ with $\delta > 0$, denote $\Delta := (\mathcal{G}_\mathrm{q}, V, M, \boldsymbol{U})$ for brevity. We call $\Phi_\Delta : \mathbb{N}^M \times \mathbb{R}_\mu \supseteq\!\!\rightarrow \{\boldsymbol{U}\}$ a *restricted-input* QGCE function if, all pairs $(\boldsymbol{y}, \lambda)$ that satisfy $\lambda \in \mathbb{R}_\mu$ and $y \in \{1, \ldots, |\mathcal{G}_{\mathrm{q},\lambda}|\}^M$, we have $\|\mathcal{G}_{\mathrm{q},\lambda}(\boldsymbol{y}) - \Phi_\Delta(\boldsymbol{y}, \lambda)\| < \delta$. Fig. 4 shows a schematics of such functions. We obtain the following.

Theorem 2 (Restricted-Input QGCE Functions): Given a possibly empty gate set $\mathcal{G}_\mathrm{q} = \{U_1, \ldots, U_N\}$, $N \in \mathbb{N}$, any non-identity $V \in \mathcal{U}_\mu$, $M \in \mathbb{N}$, and any $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}_\mu^*$ with $\delta > 0$, consider $\Delta := (\mathcal{G}_\mathrm{q}, V, M, \boldsymbol{U})$ and let $\Phi_\Delta : \mathbb{N}^M \times \mathbb{R}_\mu \supseteq\!\!\rightarrow \{\boldsymbol{U}\}$ be a restricted-input QGCE function. Then, Φ_Δ is not Turing computable.

For the proof of Theorems 1 and 2, we employ several results and techniques established in [13]. In the following, define

$$\|A\|_\mathrm{tr} := \sqrt{\mathrm{tr}\left(A^\mathrm{H} A\right)} = \left(\sum_{m,k=1}^{\dim \mathcal{H}} \left|\langle m|A|k\rangle\right|^2\right)^{\frac{1}{2}} \tag{4}$$

for all $A \in \mathbb{C}^{N\times N}$. Then, for all $A \in \mathbb{C}^{N\times N}$, we have $0 \leq \|A\| \leq \|A\|_\mathrm{tr}$. Further, for all $A \in \mathbb{C}^{N\times N}$, we have $0 = \|A\| = \|A\|_\mathrm{tr}$ if and only if we also have $A = \mathbf{0}$. Finally, observe that $A \mapsto \|A\|_\mathrm{tr}$ is $\|\cdot\|$-continuous and the mapping $(U, V) \mapsto \|U - V\|_\mathrm{tr} \in \mathbb{R}_\mu$ is computable with domain $\mathcal{U}_\mu \times \mathcal{U}_\mu$.

Proof of Theorem 1: The Theorem follows from Lemmas 2 and 3 by contradiction. Thus, in the following, assume that $\Phi_{M,\boldsymbol{U}} : \mathbb{N}^M \times \mathrm{P}\mathcal{U}_\mu \supseteq\!\!\rightarrow \{\boldsymbol{U}\}$ is total and Turing computable.

Choose $U \in \boldsymbol{U}$ and $V \in \boldsymbol{U}$ such that $\|U - V\| > \delta$. For any (possibly empty) gate set $\mathcal{G}_\mathrm{q}$, define the function $f : \mathbb{R}_\mu \cap [0,1] \mapsto \mathbb{R}_\mu,\ \lambda \mapsto f(\lambda)$ according to

$$f(\lambda) := \Big\| \Phi_{M,\boldsymbol{U}}\left(u, \mathcal{G}_\mathrm{q} \cup \left\{U\right\}\right) \ \ldots \\ \ldots \ - \Phi_{M,\boldsymbol{U}}\left(u, \mathcal{G}_\mathrm{q} \cup \left\{U^{1-\lambda}V^\lambda\right\}\right) \Big\|_\mathrm{tr},$$

where we choose $u \in \mathbb{N}$ such that we have

$$\left[\mathcal{G}_\mathrm{q} \cup \left\{U^{1-\lambda}V^\lambda\right\}\right](u) = U^{1-\lambda}V^\lambda$$

for all $\lambda \in \mathbb{R}_\mu \cap [0,1]$. We observe the following.

1) Since $\Phi_{M,\boldsymbol{U}}$ is defined for arbitrary arguments $\mathcal{G}_\mathrm{q}$, $\mathcal{G}_\mathrm{q} \cup \left\{U^{1-\lambda}V^\lambda\right\}$ is, for all $\lambda \in \mathbb{R}_\mu \cap [0,1]$, a valid argument for $\Phi_{M,\boldsymbol{U}}$ as well.
2) By choice of U and V, we have $f(0) = 0$ and $f(1) > 0$.
3) Since $\boldsymbol{U}$ is finite, the set $\mathcal{F} := \{f(\lambda) : \lambda \in \mathbb{R}_\mu \cap [0,1]\}$, i.e., the set of values attained by f for $\lambda \in \mathbb{R}_\mu \cap [0,1]$ is finite.
4) By the theorem's assumption, (4) including the subsequent consideration, and Lemma 1, it follows that f is a (total) Turing computable function.

Finally, choose $x \in \mathbb{R}_\mu$ such that $0 < x < f(1)$ and $x \notin \mathcal{F}$ hold true and define the function $F : \mathbb{R}_\mu \cap [0,1] \to \{0,1\}, \lambda \mapsto F(\lambda)$ according to

$$F(\lambda) := \mathbb{1}_{<x}(f(\lambda)).$$

Then, employing Lemma 3, F is a composition of Turing-computable functions, and therefore a Turing-computable function itself. Further, observe that $\mathrm{D}(\mathbb{1}_{<x}) = \mathbb{R}_\mu \setminus \{x\}$ and thus, by choice of x, $\mathcal{F} \in \mathrm{D}(\mathbb{1}_{<x})$. Thus, F is total. Finally, observe that $F(0) = 0$ and $F(1) = 1$. Thus, F satisfies all requirements of Lemma 2. At the same time, we have shown F to be Turing computable, contradicting Lemma 2. □

Proof of Theorem 2: The proof follows much along the same lines as the proof of Theorem 1. In the following, assume that $\Phi_\Delta : \mathbb{N}^M \times \mathbb{R}_\mu \supseteq\!\!\rightarrow \{\boldsymbol{U}\}$ is total and Turing computable.

Choose $u \in \mathbb{N}$ such that $\mathcal{G}_{\mathrm{q},\lambda}(u) = V^\lambda$ for all $\lambda \in \mathbb{R}_\mu$. Since $\boldsymbol{U}$ is a δ-net, we have $\|\mathrm{Id}_\mathcal{H} - \Phi_\Delta(u,0)\| < \delta$. Further, observe that we can choose $\eta \in \mathbb{R}_\mu$ such that V^η has eigenvalue -1, in which case we have $\|\mathrm{Id}_\mathcal{H} - \Phi_\Delta(u,\eta)\| = 2$. Define the function $f : \mathbb{R}_\mu \cap [0,1] \mapsto \mathbb{R}_\mu,\ \lambda \mapsto f(\lambda)$ according to

$$f(\lambda) := \Big\| \Phi_\Delta(u,0) - \Phi_\Delta(u,\lambda\eta) \Big\|_{\mathrm{tr}},$$

for all $\lambda \in \mathbb{R}_\mu \cap [0,1]$. We observe the following.

1) By choice of $\eta \in \mathbb{R}_\mu$, we have $f(0) = 0$ and $f(1) > 0$.
2) Since $\boldsymbol{U}$ is finite, the set $\mathcal{F} := \{f(\lambda) : \lambda \in \mathbb{R}_\mu \cap [0,1]\}$, i.e., the set of values attained by f for $\lambda \in \mathbb{R}_\mu \cap [0,1]$ is finite.
3) By the theorem's assumption, (4) including the subsequent consideration, and the effective closedness of $\mathbb{R}_\mu$ with respect to its field operations, it follows that f is a (total) Turing computable function.

Finally, choose $x \in \mathbb{R}_\mu$ such that $0 < x < f(1)$ and $x \notin \mathcal{F}$ hold true and define the function $F : \mathbb{R}_\mu \cap [0,1] \to \{0,1\}, \lambda \mapsto F(\lambda)$ according to

$$F(\lambda) := \mathbb{1}_{<x}(f(\lambda)).$$

Then, employing Lemma 3, F is a composition of Turing-computable functions, and therefore a Turing-computable function itself. Further, observe that $\mathrm{D}(\mathbb{1}_{<x}) = \mathbb{R}_\mu \setminus \{x\}$ and thus, by choice of x, $\mathcal{F} \in \mathrm{D}(\mathbb{1}_{<x})$. Thus, F is total. Finally, observe that $F(0) = 0$ and $F(1) = 1$. Thus, F satisfies all requirements of Lemma 2. At the same time, we have shown F to be Turing computable, contradicting Lemma 2. □

VI. The Uncomputability of Quantum Gate-Circuit Concatenation Functions

Subsequently, we consider general QGCC functions as addressed in Problem 2. That is, given $\delta > 0$, we consider partial functions $\Psi_\delta : \mathbb{N} \times \mathbb{N} \times (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*) \supseteq\!\!\rightarrow \mathcal{U}_\mu$ such that for all tuples $(u, v, \boldsymbol{U})$ that satisfy $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}_\mu^*$, $u \in \{1, \ldots, |\boldsymbol{U}|\}$, and $v \in \{1, \ldots, |\boldsymbol{U}|\}$, we have $\Psi_\delta(u, v, \boldsymbol{U}) \in \{\boldsymbol{U}\}$ as well as $\|\boldsymbol{U}(u)\boldsymbol{U}(v) - \Psi_\delta(u, v, \boldsymbol{U})\| < \delta$. We obtain the following.

Theorem 3 (General QGCC Functions): Given $\delta > 0$, let $\Psi_\delta : \mathbb{N} \times \mathbb{N} \times (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*) \supseteq\!\!\rightarrow \mathcal{U}_\mu$ be a general QGCC function. Then, Ψ_δ is not Turing computable.

In Section V, we have discussed the algorithmic solvability of the classical gate-circuit emulation problem using an enumeration argument. From an analogous line of reasoning, it follows that the classical equivalents of (general) QGCC functions are equally Turing computable

Analogous to Theorem 1 for (general) QGCE functions, Theorem 3 concerns QGCC functions functions defined for arbitrary arguments $\boldsymbol{U} \in \mathcal{D}_\delta \cap \mathcal{U}_\mu^*$ as shown in Fig. 3. Again, we provide a stricter version of Theorem 3 that fixes the input up to one parameter $\lambda \in \mathbb{R}_\mu$. Given a gate set $\mathcal{G}_\mathrm{q} = \{U_1, \ldots, U_N\} \subset \mathcal{U}_\mu$, $N \in \mathbb{N}$, denote by $\mathcal{G}_\mathrm{q}^\sim$ the group generated by $\mathcal{G}_\mathrm{q}$. In the following, we require that $-\mathrm{Id}_\mathcal{H}$ is an element of the closure of

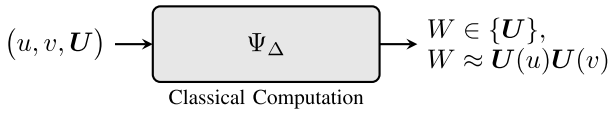

Fig. 5. Schematic depiction of restricted-input quantum gate-circut emulation functions with $\Delta := (\delta, \mathcal{G}_\mathrm{q}, V)$ and $\boldsymbol{U} \in \mathcal{D}_\delta(\mathcal{G}_\mathrm{q}, V)$, as considered in Theorem 4.

$\mathcal{G}_\mathrm{q}^\sim$. For any non-identity $V \in \mathcal{U}_\mu$, we again consider parameterized gate sets $\mathcal{G}_{\mathrm{q},\lambda} := \{U_1, \ldots, U_N, V^\lambda\}$, $\lambda \in \mathbb{R}_\mu$, denoting by $\mathcal{G}_{\mathrm{q},\lambda}^\sim$ the group generated by $\mathcal{G}_{\mathrm{q},\lambda}$. Further, we define

$$\mathcal{D}_\delta(\mathcal{G}_\mathrm{q}, V) := \bigcup\nolimits_{\lambda \in \mathbb{R}_\mu} \Big\{ \boldsymbol{U} \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*) : \{\boldsymbol{U}\} \subset \mathcal{G}_{\mathrm{q},\lambda}^\sim \Big\}$$

Then, given $\delta > 0$ and denoting $\Delta := (\delta, \mathcal{G}_\mathrm{q}, V)$ for brevity, we call $\Psi_\Delta : \mathbb{N} \times \mathbb{N} \times \mathcal{D}_\delta(\mathcal{G}_\mathrm{q}, V) \supseteq\!\!\rightarrow \mathcal{U}_\mu$ a *restricted-input* QGCC function if, for all tuples $(u, v, \boldsymbol{U})$ that satisfy $\boldsymbol{U} \in \mathcal{D}_\delta(\mathcal{G}_\mathrm{q}, V)$, $u \in \{1, \ldots, |\boldsymbol{U}|\}$, and $v \in \{1, \ldots, |\boldsymbol{U}|\}$, we have $\Psi_\Delta(u, v, \boldsymbol{U}) \in \{\boldsymbol{U}\}$ as well as $\|\boldsymbol{U}(u)\boldsymbol{U}(v) - \Psi_\delta(u, v, \boldsymbol{U})\| < \delta$. Fig. 5 shows a schematics of such functions. We obtain the following.

Theorem 4 (Restricted-Input QGCC Functions): For $\delta > 0$, $\mathcal{G}_\mathrm{q} \subset \mathcal{U}_\mu$, and $V \in \mathcal{U}_\mu$ as above, denote $\Delta := (\delta, \mathcal{G}_\mathrm{q}, V)$ and let $\Psi_\Delta : \mathbb{N} \times \mathbb{N} \times \mathcal{D}_\delta(\mathcal{G}_\mathrm{q}, V) \supseteq\!\!\rightarrow \mathcal{U}_\mu$ be a restricted-input QGCC function. Then, Ψ_Δ is not Turing computable.

In the following, we will establish the proof of Theorems 3 and 4. For $U \in \mathcal{U}_\mu$, we define $U\boldsymbol{U} := (UU_1, \ldots, UU_N)$. Observe that if $\boldsymbol{U}$ is a δ-net, then, $U\boldsymbol{U}$ is a δ-net for all $U \in \mathcal{U}$, as well, which is due to the group structure of $\mathcal{U}$ and the invariance of $\|\cdot\|$ with respect to matrix multiplication by an element of $\mathcal{U}$. Precisely, let $V \in \mathcal{U}_\mu$ be arbitrary. Since $\boldsymbol{U}$ is a δ-net by assumption, there exists $n \in \{1, \ldots, |\boldsymbol{U}|\}$ such that $\|U^{-1}V - \boldsymbol{U}(n)\| < \delta$ is satisfied. Thus,

$$\begin{aligned} &\big\|U^{-1}V - \boldsymbol{U}(n)\big\| \ \ldots \\ &\quad \ldots \ = \big\|U\big(U^{-1}V - \boldsymbol{U}(n)\big)\big\| = \big\|V - U(\boldsymbol{U}(n))\big\| \end{aligned}$$

with $U(\boldsymbol{U}(n)) = [U\boldsymbol{U}](n) \in U\boldsymbol{U}$. Since V is assumed arbitrary, $U\boldsymbol{U}$ must be a δ-net as well.

In order to prove Theorem 3, we will introduce the following preliminary Lemma. For $\delta > 0$, let

$$\Psi_\delta : \mathbb{N} \times \mathbb{N} \times (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*) \supseteq\!\!\rightarrow \mathcal{U}_\mu$$

be a general QGCC function. For $\boldsymbol{U} = (U_1, \ldots, U_N) \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*)$, and $u, v \in \{1, \ldots, |\boldsymbol{U}|\}$, we define the function

$$\Psi_\delta(\cdot|\boldsymbol{U}, u, v) : \mathcal{U}_\mu \supseteq\!\!\rightarrow \mathcal{U}_\mu,\ U \mapsto \Psi_\delta(U|\boldsymbol{U}, u, v)$$

according to

$$\Psi_\delta(U|\boldsymbol{U}, u, v) := \Psi_\delta\big(u, v, U\boldsymbol{U}\big).$$

Lemma 4: For $\delta > 0$, let $\Psi_\delta : \mathbb{N} \times \mathbb{N} \times (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*) \supseteq\!\!\rightarrow \mathcal{U}_\mu$ be a general QGCC function, and let $(U_1, \ldots, U_N) = \boldsymbol{U}$ be a δ-net. For all $u, v \in \{1, \ldots, |\boldsymbol{U}|\}$ with $\boldsymbol{U}(u) \neq \boldsymbol{U}(v)$, there exists $U \in \mathcal{U}_\mu$ such that we have

$$\Psi_\delta(U|\boldsymbol{U}, u, v) \neq U\Psi_\delta(u, v, \boldsymbol{U}). \tag{5}$$

In other words, for every triple $(u, v, \boldsymbol{U})$ such that we have $\boldsymbol{U} \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*)$, $u, v \in \{1, \ldots, |\boldsymbol{U}|\}$, there exists $U \in \mathcal{U}_\mu$ such

that applying the function Ψ_δ to $(u, v, \boldsymbol{U})$ does not commute with multiplying $(u, v, \boldsymbol{U})$ by U from the left.

Proof of Lemma 4: We prove the statement by contradiction. To this end, assume $\Psi_\delta(U|\boldsymbol{U}, u, v) = U\Psi_\delta(u, v, \boldsymbol{U})$ holds true for all $U \in \mathcal{U}_\mu$. If this is the case, we have

$$\begin{aligned}\delta &> \big\|U\boldsymbol{U}(u)U\boldsymbol{U}(v) - U\Psi_\delta(u, v, \boldsymbol{U})\big\| \\ &= \big\|\boldsymbol{U}(u)U\boldsymbol{U}(v) - \Psi_\delta(u, v, \boldsymbol{U})\big\|\end{aligned}$$

which again is due to the invariance of $\|\cdot\|$ with respect to matrix multiplication by an element of $\mathcal{U}$. Next, choose $m \in \{1, \dots, N\}$ arbitrary and set $U := \boldsymbol{U}(m)(\boldsymbol{U}(v))^{-1}$. Then, we obtain $\delta > \big\|\boldsymbol{U}(u)\boldsymbol{U}(m) - \Psi_\delta(u, v, \boldsymbol{U})\big\|$, and, employing the triangle inequality subsequently,

$$\begin{aligned}&\big\|\boldsymbol{U}(u)\boldsymbol{U}(m) - \boldsymbol{U}(u)U[-b](v)\big\| \ \dots \\ &\quad \dots \ \leq \big\|\boldsymbol{U}(u)\boldsymbol{U}(m) - \Psi_\delta(u, v, \boldsymbol{U})\big\| \ \dots \\ &\qquad \dots \ + \big\|\boldsymbol{U}(u)\boldsymbol{U}(v) - \Psi_\delta(u, v, \boldsymbol{U})\big\| \leq 2\delta\end{aligned}$$

for all $m \in \{1, \dots, |\boldsymbol{U}|\}$. Employing the invariance of $\|\cdot\|$ with respect to matrix multiplication by an element of $\mathcal{U}$ once more, we have

$$\big\|\boldsymbol{U}(m) - \boldsymbol{U}(v)\big\| \leq \big\|\boldsymbol{U}(u)\boldsymbol{U}(m) - \boldsymbol{U}(u)\boldsymbol{U}(v)\big\| \leq 2\delta$$

for all $1 \leq m \leq N$. Finally, choose $V \in \mathcal{U}_\mu$ and $m \in \{1, \leq, |\boldsymbol{U}|\}$ such that $\|V - \boldsymbol{U}(v)\| = 2$ and $\|V - \boldsymbol{U}(m)\big\| < \delta$ are satisfied. By further application of the triangle inequality, we obtain

$$2 = \|V - \boldsymbol{U}(v)\| \leq \|\boldsymbol{U}(m) - \boldsymbol{U}(v)\big\| + \delta < 3\delta,$$

which is a contradiction. Since $u, v \in \{1, \dots, |\boldsymbol{U}|\}$ were chosen arbitrarily, the claim follows. □

Proof of Theorem 3: We prove the theorem by contradiction. To this end, consider $\delta \in \mathbb{R}$ with $\delta \geq 0$ and assume Ψ_δ to be a Turing-computable general QGCC function. For the sake of a clear structure, we divide the proof into paragraphs.

§1. According to Lemma 4, we can choose $u, v \in \{1, \dots, |\boldsymbol{U}|\}$ and $U \in \mathcal{U}_\mu$ such that (5) is satisfied. Define the function $\psi : \mathbb{R}_\mu \cap [0, 1] \to \mathcal{U}_\mu,\ \lambda \mapsto \psi(\lambda)$, according to

$$\psi(\lambda) := \Psi_\delta(U^\lambda|\boldsymbol{U}, u, v)$$

for all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$. Combined with Lemma 1, the assumption of Ψ_δ being Turing computable implies that ψ is Turing computable as well.

§2. Observe that we have $\psi(0) = \Psi_\delta(u, v, \boldsymbol{U})$. Thus, for all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$, we have $U^\lambda\psi(0) = U^\lambda\Psi_\delta(u, v, \boldsymbol{U})$. We define the total function $F : \mathbb{R}_\mu \cap [0, 1] \to \mathbb{R}_\mu,\ \lambda \mapsto F(\lambda)$ according to

$$\begin{aligned}F(\lambda) :&= \big\|\Psi_\delta\big(U^\lambda|\boldsymbol{U}, u, v\big) - U^\lambda\Psi_\delta(u, v, \boldsymbol{U})\big\|_{\text{tr}} \\ &= \big\|\psi(\lambda) - U^\lambda\psi(0)\big\|_{\text{tr}}\end{aligned}$$

for all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$. Further, we observe the following for F, $\psi(\lambda)$, $U^\lambda\psi(0)$, and all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$.

1) Employing (4) in combination with Lemma 1, it follows that F is a Turing-computable function.
2) For all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$, we have $\psi(\lambda) \in U^\lambda\boldsymbol{U}$ as well as $U^\lambda\psi(0) \in U^\lambda\boldsymbol{U}$.
3) We have $\psi(\lambda) = U^\lambda\psi(0)$ for $\lambda = 0$, while we have $\psi(\lambda) \neq U^\lambda\psi(0)$ for $\lambda = 1$.

§3. Recall that $(U_1, \dots, U_N) = \boldsymbol{U} \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*)$ is finite per definition. Hence, defining

$$\begin{aligned}\epsilon(\boldsymbol{U}) := \min\Big\{&\|\boldsymbol{U}(u) - U[-b](v)\|_{\text{tr}} : \ u, v \in \{1, \dots, |\boldsymbol{U}|\} \ \dots \\ &\dots \text{ with } \boldsymbol{U}(u) \neq \boldsymbol{U}(v)\Big\},\end{aligned}$$

we have $\epsilon(\boldsymbol{U}) > 0$ for all $\boldsymbol{U} \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*)$ with more than one distinct component. From (4), it follows that – just as $\|\cdot\|, -\|\cdot\|_{\text{tr}}$ is invariant with respect to matrix multiplication by an element of $\mathcal{U}$. Accordingly, for all $\boldsymbol{U} \in (\mathcal{D}_\delta \cap \mathcal{U}_\mu^*)$ with more than one distinct component, all $U \in \mathcal{U}_\mu$, and all $u, v \in \{1, \dots, |\boldsymbol{U}|\}$, we have

$$\|U\boldsymbol{U}(u) - U\boldsymbol{U}(v)\|_{\text{tr}} = \|U\boldsymbol{U}(u) - U\boldsymbol{U}(v)\|_{\text{tr}}.$$

Consequently, we have $\epsilon(U^\lambda\boldsymbol{U}) = \epsilon(\boldsymbol{U})$ for all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$.

§4. For all $\lambda \in \mathbb{R}_\mu \cap [0, 1]$, it follows from §1 and §2 that we have

$$F(\lambda) \in (\mathbb{R}_\mu \cap [\epsilon(\boldsymbol{U}), \delta]) \cup \{0\},$$

with $F(0) = 0$ and $F(1) \geq \epsilon(\boldsymbol{U}) > 0$. Choose any $x \in \mathbb{R}_\mu$ that satisfies $0 < x < \epsilon(\boldsymbol{U})$ and recall that F is a Turing-computable function. Then, as follows from Lemma 3,

$$\begin{aligned}\mathbb{1}_{<x} \circ F : \mathbb{R}_\mu \cap [0, 1] &\to \{0, 1\}, \\ \lambda &\mapsto [\mathbb{1}_{<x} \circ F](\lambda) := \mathbb{1}_{<x}(F(\lambda))\end{aligned}$$

is a composition of Turing-computable functions that satisfies $\mathrm{D}(\mathbb{1}_{<x} \circ F) = \mathbb{R}_\mu \cap [0, 1]$, $[\mathbb{1}_{<x} \circ F](0) = 0$, and $[\mathbb{1}_{<x} \circ F](1) = 1$. Accordingly, $\mathbb{1}_{<x} \circ F$ is a Turing-computable function that violates Lemma 2, which concludes the proof. □

Let $\Psi_\Delta : \mathbb{N} \times \mathbb{N} \times \mathcal{D}_\delta(\mathcal{G}_\text{q}, V) \supseteq\!\to \mathcal{U}_\mu$ be a restricted-input QGCC function. For $\boldsymbol{U} = (U_1, \dots, U_N) \in \mathcal{D}_\delta(\mathcal{G}_\text{q}, V)$ and $u, v \in \{1, \dots, |\boldsymbol{U}|\}$, we define the function

$$\Psi_\Delta(\cdot|\boldsymbol{U}, u, v) : \mathcal{U}_\mu \supseteq\!\to \mathcal{U}_\mu,\ U \mapsto \Psi_\delta(U|\boldsymbol{U}, u, v)$$

according to

$$\Psi_\Delta(U|\boldsymbol{U}, u, v) := \Psi_\Delta\big(u, v, U\boldsymbol{U}\big),$$

where $\Psi_\Delta(U|\boldsymbol{U}, u, v)$ is well-defined whenever $U\boldsymbol{U}$ is a feasible input for Ψ_Δ.

Proof of Theorem 4: The proof follows much along the same lines as the proof of Theorem 3, with the exception that Lemma 4 is not applicable directly. Subsequently, assume Ψ_Δ Turing computable. In the proof of Theorem 3, substitute Ψ_Δ for Ψ_δ and replace §1 as follows.

§1. By assumption, there exists $\boldsymbol{U} \in \mathcal{D}_\delta(\mathcal{G}_\text{q}, V)$ with $u, v \in \{\boldsymbol{U}\}$ such that $\|\mathrm{Id}_\mathcal{H} + \boldsymbol{U}(u)\| < \delta$ and $\|\mathrm{Id}_\mathcal{H} + \boldsymbol{U}(v)\| < \delta$ are satisfied, in which case we have $\|\mathrm{Id}_\mathcal{H} - \boldsymbol{U}(u)\boldsymbol{U}(v)\| < 2\delta$. Accordingly, we have

$$\|U^2 - \Psi_\Delta(U|\boldsymbol{U}, u, v)\| < \eta_1, \quad \|U - U\Psi_\Delta(u, v, \boldsymbol{U})\| < \eta_2$$

for some small $\eta_1, \eta_2 > 0$. Choose γ such that V^γ has eigenvalue -1 and set $U := V^\gamma$. Then, we have $\|U - U^2\| = 2$, and thus

$$\|\Psi_\Delta(U|\boldsymbol{U}, u, v) - U\Psi_\Delta(u, v, \boldsymbol{U})\| > 2 - (\eta_1 + \eta_2).$$

Accordingly, we have $\Psi_\Delta(U|\boldsymbol{U},u,v) \neq U\Psi_\Delta(u,v,\boldsymbol{U})$, and $(u,v,U^\lambda\boldsymbol{U})$ is a feasible input for Ψ_Δ for all $\lambda \in \mathbb{R}_\mu \cap [0,1]$. Next, we define the function $\psi : \mathbb{R}_\mu \cap [0,1] \to \mathcal{U}_\mu,\ \lambda \mapsto \psi(\lambda)$, according to $\psi(\lambda) := \Psi_\Delta(U^\lambda|\boldsymbol{U},u,v)$ for all $\lambda \in \mathbb{R}_\mu \cap [0,1]$. Combined with Lemma 1, the assumption of Ψ_Δ being Turing computable implies that ψ is Turing computable as well.

The further derivation of the statement equals §2, §3 and §4 of the proof of Theorem 3. □

VII. Conclusion

In the present work, we have discussed the computability of QGCE and QGCC functions, with Problems 1 and 2 formalizing the relevant research questions. Following Theorems 1, 2, 3 and 4, we answer both questions to the negative: Neither QGCE nor QGCC functions are Turing computable. This is in direct contrast to gate-circuit emulation and concatenation in the bit-string model of classical computing. Since the set of Boolean functions is enumerable and the relevant equality predicate is recursive, classical gate-circuit emulation and concatenation is computable even for varying circuit depth and width.

Observe that for any general QGCE function $\Phi_{M,\boldsymbol{U}}$, any general QGCC function Ψ_δ, and $\mathcal{G}_\mathrm{q} \subset \mathcal{U}$ and $\boldsymbol{U} \subset \mathcal{U}$ arbitrary but fixed, the functions $\boldsymbol{y} \mapsto \Phi_{M,\boldsymbol{U}}(\boldsymbol{y},\mathcal{G}_\mathrm{q})$ and $(u,v) \mapsto \Psi_\delta(u,v,\boldsymbol{U})$ essentially implement finite lookup tables, and are thus always Turing computable. In contrast, $\Phi_{M,\boldsymbol{U}}$ and Ψ_δ are defined for (varying) arguments $\mathcal{G}_\mathrm{q} \subset \mathcal{U}$ and $\boldsymbol{U} \subset \mathcal{U}$. In other words, any algorithmic implementation of $\Phi_{M,\boldsymbol{U}}$ or Ψ_δ would have to compute the relevant lookup table automatically, which is impossible according to Theorems 1 and 3. Again, this is in contrast to the classical analogons of QGCE and QGCC, where it is always possible to compute the relevant lookup-tables algorithmically. As indicated in Section I, there exist multiple potential approaches to implement gate based quantum computing, which, in the practical context, require assessment with regarding their performance. If one manufacturer wants to benchmark their implementation versus those of other manufacturers, they must solve an emulation problem. as follows from our results, the manufacturer has to implement a suitable lookup table "by hand", which is an engineering rather than a computer science problem. As put forward in [28], "*The fundamental question underlying all computing is 'What can be (efficiently) automated'?*". In our case, this would imply the use of an algorithm that implements a quantum gate-circuit emulation function. Our results show that this is an unsolvable task. The non-automated design of lookup tables "by hand" – that is, in the tradition of engineering – is the only feasible solution.

Generally, one may ask for which subsets of $\mathrm{P}\mathcal{U}_\mu$ and $\mathcal{D}_\delta \cap \mathcal{U}_\mu^*$ the resulting restricted-input QGCE and QGCC problems are computable. The list of relevant subsets includes, for example

1) the set of unitary matrices whose entries have rational-valued real and imaginary parts;
2) the set of unitary matrices characterized by a Hamiltonian whose entries have real-valued real and imaginary parts;
3) the set of unitary matrices whose entries are *elementary numbers* [29], [30].

To the best of the authors' knowledge, the question of whether the restricted-input QGCE and QGCC problems for these sets are computable is open. However, the results presented in this article have implications for the structural properties that subsets of $\mathrm{P}\mathcal{U}_\mu$ and $\mathcal{D}_\delta \cap \mathcal{U}_\mu^*$ for which the resulting restricted-input QGCE and QGCC problems are computable have to satisfy. According to Theorems 2, and 4, they may not contain non-trivial smooth one-dimensional manifolds. Further, they will generally not satisfy asymptotic closedness in the sense of computable analysis.

As indicated before, [13], [31] recently showed the non-existence of Turing-computable gate-circuit *compiler* functions. For a fixed universal gate set, functions of this type map arbitrary quantum algorithms to an approximating gate circuit such that the resulting approximation error is smaller than a prescribed margin. Together with [13], [31], the present paper forms a coherent picture of the fundamental limits of gate-based quantum computing.

In a broader context, the present paper relates to recent results on how the non-existence of Turing-computable solutions to mathematical problems affects the satisfiability of legal requirements in algorithm design [32]. Among others, algorithmic transparency and the right-to-explainability may be compromised when attempting to implement a solution to such problems on digital hardware. In the scope of quantum information processing, adherence to legal requirements is relevant for, e.g., future machine-learning technologies or the trustworthiness of communication and computing systems. For comprehensive discussions, we refer the reader to [33] (trustworthiness requirements for near-future communication technologies) and [34] (quantum computing in the context of networked systems). Whether the results of the present paper in conjunction with [13], [31] entail legal consequences for gate-based quantum computing similar to those discussed in [32] remains a pressing open research question.

On a similar note, one may ask whether the methods employed in the present paper and [13], [31] remain valid for the restricted case of *quantum variational algorithms* (QVAs). If, in addition, the results on Turing computability that [35], [36] established transfer to QVAs, gate-based quantum computers are not generally able to implement QVAs under strict performance and error-tolerance requirements. It is still to be determined if, in this case, legal implications such as those discussed in [32] analogously apply to QVAs.

Finally, the authors want to highlight that for many practically relevant applications, particularly in the field of *information and communication technology* (ICT), *analog* quantum computing may provide feasible hardware acceleration techniques. For an extensive discussion, we refer to [37].

Acknowledgment

Holger Boche thanks Eike Kiltz from the DFG Cluster of Excellence CASA for discussions on quantum computing

algorithms for cryptography. Holger Boche and Yannik Böck also thank Immanuel Bloch from the DFG Cluster of Excellence MCQST for discussions on quantum computing approaches with continuous parameter sets.

REFERENCES

[1] Y. N. Böck, H. Boche, Z. G. del Toro, and F. H. P. Fitzek, "Feynman meets Turing: The uncomputability of quantum gate-circuit emulation and concatenation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2024, pp. 2062–2067.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," *Proc. R. Soc. London A Math. Phys. Sci.*, vol. 400, no. 1818, pp. 97–117, Jul. 1985.

[13] 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.

[14] J. Nolt, A. Varzi, and D. Rohatyn, *Schaum's Outline of Theory and Problems of Logic*, 2nd ed. New York, NY, USA: McGraw-Hill Education, 1998.

[15] H. B. Enderton, *A Mathematical Introduction to Logic*. Academic Press, 2001.

[16] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Elect. Eng.*, vol. 57, no. 12, pp. 713–723, 1938.

[17] S. A. Cook, *Computational Complexity of Higher Type Functions*. AMS, 1990.

[18] L. A. Levin, "Universal search problems," pp. 399–400, 1984.

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

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

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

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

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

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

[25] H. Boche, Y. N. Böck, U. J. Mönich, and F. H. P. Fitzek, "Trustworthy digital representations of analog information – An application-guided analysis of a fundamental theoretical problem in digital twinning," *Algorithms*, vol. 16, no. 11, 2023, Art. no. 514.

[26] A. Bauer and D. S. Scott, "The realizability approach to computable analysis and topology," Ph.D. thesis, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-00-164, Sep. 2000, p. 248.

[27] H. Boche, R. F. Schaefer, and H. V. Poor, "Turing meets Shannon: On the algorithmic construction of channel-aware codes," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2256–2267, Apr. 2022.

[28] D. E. Comer et al., "Computing as a discipline," *Commun. ACM*, vol. 32, no. 1, pp. 9–23, Jan. 1989.

[29] D. Richardson, "How to recognize zero," *J. Symbolic Comput.*, vol. 24, no. 6, pp. 627–645, 1997.

[30] T. Y. Chow, "What is a closed-form number?" *Amer. Math. Monthly*, vol. 106, no. 5, pp. 440–448, 1999.

[31] Y. N. Böck, H. Boche, Z. Garcia 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)*, Denver, CO, USA, 2024, pp. 2440–2445.

[32] H. Boche, A. Fono, and G. Kutyniok, "A mathematical framework for computability aspects of algorithmic transparency," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2024, pp. 3089–3094.

[33] G. P. Fettweis and H. Boche, "On 6G and trustworthiness," *Commun. ACM*, vol. 65, no. 4, pp. 48–49, 2022.

[34] R. Bassoli et al., *Quantum Communication Networks* (Foundations in Signal Processing Communications and Networking), vol. 23. Springer-Verlag, 2021.

[35] H. Boche, R. F. Schaefer, and H. V. Poor, "Algorithmic computability and approximability of capacity-achieving input distributions," *IEEE Trans. Inf. Theory*, vol. 69, no. 9, pp. 5449–5462, Sep. 2023.

[36] Y. Lee, H. Boche, and G. Kutyniok, "Computability of optimizers," *IEEE Trans. Inf. Theory*, vol. 70, no. 4, pp. 2967–2983, Apr. 2024.

[37] Y. Böck, H. Boche, R. Bassoli, and F. H. Fitzek, "Foundations of in-network quantum computing for future communication networks," in *Proc. 33rd Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2024, pp. 1–9.

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. Currently, he is 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.

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

Zoe Garcia del Toro is a Student Research Assistant with the Chair of Theoretical Information Technology, Technical University of Munich, Munich, Germany. She is currently working toward the M.Sc. degree in quantum science and technology program from the Technical University of Munich and Ludwig-Maximilian University, Munich, Germany. Her research focuses on applications of computability theory and quantum information theory.

Frank H. P. Fitzek (Senior Member, IEEE) received the diploma degree in electrical engineering from RWTH Aachen, Germany, in 1997, and the Ph.D. degree in electrical engineering from the Technical University Berlin, in 2002. He is a Professor and the Chair of the "Deutsche Telekom Chair of Communication Networks", TU Dresden, leads at 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. In 2023, 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. His research ambitiously spans 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.