Computability of Matrix Functions and Compiling of Matrix Problems on Quantum Computers

1st Holger Boche
TUM School of Computation, Information and Technology
Technical University of Munich, Germany,
boche@tum.de

2nd Adalbert Fono
Department of Mathematics
Ludwig-Maximilians-Universität
München, Germany
fono@math.lmu.de

3rd Gitta Kutyniok
Department of Mathematics
Ludwig-Maximilians-Universität
München, Germany
kutyniok@math.lmu.de

***Abstract*—We study quantum circuit synthesis through the lens of quantum compiler functions, i.e., mappings from unitary matrices to gate-circuit approximations. While the existence of compiler functions yielding low-complexity gate-circuit approximations for arbitrary unitary matrices is well established, their implementation is inherently classical and must therefore be assessed within the framework of digital computation. Digitally computing these functions has recently been shown to be infeasible in terms of Turing computability, i.e., uniform, universally correct, and error-controlled implementations. We strengthen this limitation by proving the non-existence of computable implementations even for restricted and practically relevant sub-tasks. In particular, we establish an infeasibility result for compiler functions based on currently known quantum algorithms for estimating features of solutions to linear systems. Our approach relies on a detailed computability analysis of associated matrix functions—most notably the pseudoinverse and the condition number—within the Turing model. This yields a complete characterization of the computability breakdown and, consequently, of the infeasibility of corresponding universal and computable compiler functions.**

For a version of this work including all proofs, we refer to [1]

I. Introduction

Quantum computing aims to exploit quantum mechanical phenomena to outperform classical digital computation in specific tasks, such as integer factorization and the simulation of protein dynamics in computational chemistry [2]–[4]. However, despite substantial progress, the physical realization of scalable quantum computers remains highly challenging.

H. B. is also with the Munich Quantum Valley (MQV) and he and G. K. are with the Munich Center for Quantum Science and Technology (MCQST).

H. B., A.F., and G.K. acknowledge support by the project "Next Generation AI Computing (gAIn)", funded by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism.

H. B. was supported by the German Federal Ministry of Research, Technology and Space (BMFTR) in the programme of "Souverän. Digital. Vernetzt", research HUB 6G-life, project identification number: 16KISK002, and the BMBF Quantum Projects QUIET, Grant 16KISQ093, QD-CamNetz, Grant 16KISQ077, and QuaPhySI, Grant 16KIS1598K. He also acknowledges funding by the German Research Foundation as part of Germany's Excellence Strategy – EXC 2050/2 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden.

G. K. was supported by the Konrad Zuse School of Excellence in Reliable AI (DAAD), the Munich Center for Machine Learning (MCML), and the German Research Foundation under Grants DFG-SPP-2298, KU 1446/31-1 and KU 1446/32-1.

The most established quantum computing model, both theoretically and experimentally, is gate-based quantum computing [5]–[9]. In this framework, each gate corresponds to a unitary operator on a finite-dimensional Hilbert space and computation proceeds via sequences of quantum gates forming a circuit. Abstracting from measurement, the process separates into a quantum component—applying gates from a fixed, finite set to a state vector—and a classical component that determines and controls the gate sequence. The latter constitutes the quantum compiler problem (or circuit synthesis) of compiling quantum algorithms to gate-circuits processed by classical digital hardware [10]–[12].

Since only countably many such operators can be exactly realized by finite circuits, arbitrary unitary transformations (corresponding to quantum algorithms) must be approximated. A gate family is called universal if such approximations exist for every unitary operator with accuracy 2^{-n} for all $n \in \mathbb{N}$ [13]–[15]. While the existence of compiler functions (enabling circuit synthesis) is guaranteed, e.g., by the Solovay–Kitaev theorem [16], [17], their computable realization on digital hardware is a separate issue: a fundamental but often implicit assumption is that the classical control underlying circuit synthesis can itself be straightforwardly implemented on digital hardware. However, this is nontrivial since classical computation operates on discrete mathematical objects and does not naturally capture analytic notions such as continuity and approximation that are inherent in quantum circuit synthesis.

Indeed, recent work shows that universal compiler functions do not admit computable implementations in the sense of Turing, i.e., universally correct implementations with error control [10]–[12]. This raises the question of whether computable implementations remain feasible for restricted yet practically relevant sub-tasks. In this work, we address this question for solving systems of linear equations. Quantum algorithms for estimating features of such solutions have been proposed [18]–[20] and underpin many quantum machine learning approaches [21], [22]. A key parameter in these algorithms is the condition number of the input matrix representing the linear system. Consequently, any compiler function that maps a matrix to a quantum circuit (corresponding to these quantum algorithms) must compute the condition number as part of its classical pre-processing unless receiving it as input.

This strict requirement follows a classical definition of computer science as the science of automated problem-solving [23], meaning that for any feasible input instance—here, matrices—the computing process should yield an (approximation) of the solution—here, quantum circuits—. We show that this constitutes a fundamental obstruction: there is no universally correct, error-controlled, and uniform algorithmic implementation of the corresponding compiler function on digital hardware over sufficiently large input domains. In particular, even for fixed matrix dimensions, the computation of the condition number—and thus the associated compiler task—is not computable. This implies that, in this setting, the anticipated quantum advantage cannot be realized via provably correct classical compilation.

Overcoming this limitation would require either quantum algorithms that avoid such non-computable pre-processing or restrictions to problem instances where relevant non-computable quantities, such as the condition number, are a priori bounded (making their computation obsolete) or amenable to provably correct computation. Indeed, we also specify input domain restrictions, i.e., only consider matrices with certain properties as admissible inputs, that allow to circumvent the raised computability issues by design or, more precisely, the additional information provided by the promise influences the computability structure. However, this articulates the gap between formal non-computability in the spirit of automated problem solving and practical numerical approaches with lower requirements. In the latter, the challenge of input verification (does the given matrix actually conform to the properties specified by the input domain, in particular, if high-dimensional unstructured input instances are to be expected?) and the severity of the task restriction (does the admissible input domain still allow for sufficiently general descriptions?) are typically left open.

A. Problem Statement

The condition number of a matrix $A \in \mathbb{C}^{m\times n}$ is defined via its pseudoinverse $A^\dagger \in \mathbb{C}^{n\times m}$ [24], [25] characterized by

$$AA^\dagger A = A, \qquad A^\dagger AA^\dagger = A^\dagger,$$
$$(AA^\dagger)^H = AA^\dagger, \qquad (A^\dagger A)^H = A^\dagger A.$$

These equations uniquely define the pseudoinverse, i.e., exactly one matrix $A^\dagger$ satisfies all four conditions. While closed-form representations of the pseudoinverse are known for both universal [26]–[28] and specific matrix classes [29], their implementability on digital hardware with correctness guarantees and error control in the sense formalized in Section II remains unclear. In contrast, simple expressions arise only in special cases; e.g., if A has full column rank, then it is easy to verify that $A^\dagger = (A^H A)^{-1}A^H$.

In practice, the pseudoinverse is computed using methods such as rank decomposition, QR factorization, singular value decomposition [30], or Gaussian elimination-based approaches [31], [32]. However, these methods typically lack either full generality or provable error control in the above sense. As a result, while they often produce accurate outputs in practice and are provably reliable in restricted settings—such as well-conditioned inputs [33], [34]—their intrinsic failure modes and robustness are not fully characterized.

Thus, a universal algorithmic description enabling effective computation is missing: namely, a procedure that, given an arbitrary matrix (via converging rational approximations) and a target accuracy, computes an approximation of its pseudoinverse with target accuracy. More broadly, this raises the question of whether or under which conditions the pseudoinverse and related quantities, such as least-squares solutions and the condition number, admit effective approximation on digital hardware. To address this, we study the following mappings:

$$f^\dagger : \mathbb{C}^{m\times n} \to \mathbb{C}^{n\times m}, A \mapsto A^\dagger \tag{I}$$
$$f^{\|\dagger\|} : \mathbb{C}^{m\times n} \to \mathbb{R}, A \mapsto \|A^\dagger\|_F \tag{II}$$
$$f_{\text{lsq}} : \mathbb{C}^{m\times n} \times \mathbb{C}^m \to \mathbb{R}, (A,b) \mapsto \min_{x\in\mathbb{C}^n} \|Ax-b\|_2 \tag{III}$$
$$f_{\text{lsq-m}} : \mathbb{C}^{m\times n} \times \mathbb{C}^m \to \mathbb{C}^n, (A,b) \mapsto \operatorname*{arg\,min}_{x:\|Ax-b\|_2 = f_{\text{lsq}}(A,b)} \|x\|_2 \tag{IV}$$
$$f_{\text{lsq-n}} : \mathbb{C}^{m\times n} \times \mathbb{C}^m \to \mathbb{R}, (A,b) \mapsto \|f_{\text{lsq-m}}(A,b)\|_2 \tag{V}$$
$$\kappa : \mathbb{C}^{m\times n} \to \mathbb{R}, A \mapsto \|A\|_F \|A^\dagger\|_F \tag{VI}$$

The posed problem(s) relate to the study of effective computations (and lack thereof) in linear algebra [35]–[38] with prior works focusing on, e.g., eigenvalue computation [39].

B. Contributions

We study the existence of effective algorithms on Turing machines that approximate the mappings (I)–(VI). Since the pseudoinverse is central to all of them, we examine its effective computability for any matrix $A \in \mathbb{C}^{m\times n}$. This naturally splits into two questions: (i) for a fixed matrix A, does there exist a Turing machine TM_A that computes $A^\dagger$ with error control? and (ii) does there exist a (uniform) Turing machine TM_U that, given any A, constructs TM_A? We show that the first question admits a positive answer, whereas the second does not. Extending this negative result to the condition number yields the previously stated impossibility of a computable implementation of compiler functions for quantum algorithms solving linear systems [18] in the discussed sense. We next characterize this computability breakdown in more detail.

- We prove in Theorem 6 that no uniform algorithm, i.e., no Turing machine TM_U, can approximate the pseudoinverse with finite error: any such algorithm incurs arbitrarily large errors on certain inputs. In contrast, Proposition 9 shows that, for each fixed matrix A, a Turing machine computing $A^\dagger$ with error control does exist. Thus, the computability breakdown arises at the uniform level—specifically, in constructing TM_A from A. This construction requires additional information about A that is unavailable in this general setting, a phenomenon we illustrate for iterative methods with convergence guarantees in Corollary 8.
- These results suggest that restricting the input domain—thereby increasing available information—can restore effective computability. We confirm this for two classes of matrices, namely full-rank matrices and matrices with entries bounded away from zero, by establishing the existence

of effective algorithms in both cases. This highlights the importance of identifying structural properties of admissible inputs in practical applications.
- Finally, Theorem 11 provides a complete characterization of the computability breakdown and yields a criterion for assessing whether a specific input domain admits effective approaches. While this aligns with known discontinuities of the pseudoinverse mapping, we emphasize that no algorithm can universally decide whether a given matrix satisfies the input restrictions. Thus, no verifiably correct "red flag" mechanism exists for identifying such benign instances.

The computational barrier we identify is not a consequence of numerical errors in practical implementations but an intrinsic feature of Turing machines. While the condition number of an invertible matrix quantifies the sensitivity of inversion to numerical errors in floating-point arithmetic, it does not capture this fundamental limitation. In fact, the inverse of a fixed matrix can be computed effectively, even for ill-conditioned instances. By contrast, non-computability constitutes a strict impossibility result for any realization on digital hardware. Rather than enabling effective computation, the condition number serves only as a heuristic indicator of whether a given input may be numerically benign in specific settings.

II. Digital Computations

Modern computing is predominantly based on digital hardware. While digital systems can, in principle, compute exact solutions for discrete problems, continuous problems can only be treated approximately. This necessitates a rigorous notion of approximation to assess and guarantee correctness. A standard approach is to require effective algorithms, which produce approximations together with explicit error bounds to the (generally unknown) exact solution, and can meet any prescribed accuracy. A theoretical model of digital computation is given by Turing machines [40], which capture the logical structure of digital computation while abstracting from practical constraints such as memory or energy limitations. Any algorithm executable on real-world digital hardware can, in principle, be simulated by a Turing machine, and vice versa. Accordingly, the existence of effective algorithms for digital computation is studied within the Turing framework [41]–[47].

A. Computable Analysis

This section introduces the necessary notions from computability theory on continuous domains within the Turing framework. For comprehensive treatments of real-valued computability, see [48]–[51]; foundational topics such as recursive functions and formal definitions of Turing machines are covered by elementary textbooks [52]. The underlying paradigm, originating with Turing himself [40], models computation on real numbers via transformations of converging sequences.

Definition 1.
- *A sequence $(r_k)_{k\in\mathbb{N}}$ in $\mathbb{Q}$ is computable, if there exist recursive functions $a, b, s : \mathbb{N} \to \mathbb{N}$ such that*

$$b(k) \neq 0 \text{ and } r_k = (-1)^{s(k)} \cdot \frac{a(k)}{b(k)} \qquad \forall k \in \mathbb{N}.$$

- *A sequence $(x_k)_{k\in\mathbb{N}}$ in $\mathbb{R}$ converges effectively to $x \in \mathbb{R}$, if there exists a recursive function $e : \mathbb{N} \to \mathbb{N}$ such that*

$$|r_k - x| \leq 2^{-N} \quad \forall N \in \mathbb{N} \text{ and all } k \geq e(N).$$

- *A real number $x \in \mathbb{R}$ is computable, if there exists a computable rational sequence $(r_k)_{k\in\mathbb{N}}$ converging effectively to x. We call $(r_k)_{k\in\mathbb{N}}$ a representation of x.*
- *A sequence $(x_n)_{n\in\mathbb{N}}$ in $\mathbb{R}$ is* computable*, if there exist a computable rational sequence $(r_{n,k})_{n,k\in\mathbb{N}}$ and a recursive function $e : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that*

$$|x_n - r_{n,k}| \leq 2^{-N} \quad \forall n, N \in \mathbb{N} \text{ and all } k \geq e(n, N).$$

Remark 2. *We denote the set of computable real numbers by $\mathbb{R}_c$. Moreover, the previous definitions can be extended to $\mathbb{R}^d$ and $\mathbb{C}$—understood as a 2-dimensional real vector space—by requiring that each one-dimensional component(-wise sequence) is computable. We denote the set of computable complex numbers by $\mathbb{C}_c := \{x + iy \in \mathbb{C} : x, y \in \mathbb{R}_c\}$.*

The computability of functions is a well-studied notion with several formal variants [51]. We adopt the following concepts to derive both positive and negative results on effective computation over digital hardware.

Definition 3. *For $D \subset \mathbb{C}^d$, a function $f : D \to \mathbb{C}_c^m$ is*
- *Borel-Turing computable, if there exists a Turing machine that transforms each representation of $x \in D \cap \mathbb{C}_C^d$ into a representation for $f(x)$.*
- *Banach-Mazur computable, if f maps any computable sequences $(x_n)_{n\in\mathbb{N}}$ in $D \cap \mathbb{C}_c^d$ onto computable sequences $(f(x_n))_{n\in\mathbb{N}}$.*

Remark 4. *Borel–Turing computability captures the intuitive notion of effective computation, requiring that a function can be approximated to arbitrary accuracy with explicit error control. Thus, effective algorithms only exist for Borel–Turing computable problems. A weaker notion is Banach–Mazur computability, which requires only the preservation of computability of sequences. It is well known that Borel–Turing computability implies Banach–Mazur computability, and that functions computable in either sense are continuous [51].*

Remark 5. *Basic arithmetic operations, as well as elementary operations such as the absolute value function, are computable [50]. This also immediately implies the computability of the Euclidean and Frobenius norm.*

A slightly different computability notion is established via oracles providing representations for any—not just computable—objects [53]. For our purposes, the natural domain is characterized by computable objects, and we apply the introduced computability framework in the remainder.

III. Main Results

We now present our results on the effective computation of the mappings (I)–(VI). The central step is the computation of the pseudoinverse mapping, for which we first establish the non-existence of a uniform effective approximation. We

then analyze the breakdown of existing algorithmic strategies based on iterative methods and identify input restrictions under which effectiveness can be recovered.

A. Banach-Mazur Non-Computability of the Pseudoinverse

Borel–Turing computability of the pseudoinverse mapping would immediately imply effective computation of the functions (I)–(VI) on digital hardware. This raises the question of whether a constructive algorithm can establish computability of this mapping. A natural candidate is the singular value decomposition (SVD), a standard method for computing the pseudoinverse. However, a key step in the SVD requires identifying non-zero singular values of the input matrix, which entails comparisons to zero—an operation that cannot be performed effectively on digital hardware [50]. Thus, SVD does not yield an effective approach for computing the pseudoinverse. The following theorem shows that this limitation is not specific to SVD: in general, no effective approximation of the pseudoinverse exists.

Theorem 6. *For a function f, denote a function with the same domain and codomain by $\hat{f}$. Then, for $m, n \geq 2$, the functions (I)-(VI) are not algorithmically approximable, i.e., there exists no Banach-Mazur computable function $\hat{f}$ such that*

(i) $\sup\limits_{\substack{A\in\mathbb{C}^{m\times n}:\\ \|A\|_F\leq\sqrt{2}}} \|f(A)-\hat{f}(A)\|_F < \infty$ *for* $f \in \{f^\dagger, f^{\|\dagger\|}, \kappa\}$;

(ii) $\sup\limits_{\substack{(A,b)\in\mathbb{C}^{m\times n}\times\mathbb{C}^m:\\ \|A\|_F\leq\sqrt{2},\|b\|_2\leq\sqrt{2}}} |f(A,b)-\hat{f}(A,b)| < \frac{1}{4}$ *for* $f = f_{lsq}$;

(iii) $\sup\limits_{\substack{(A,b)\in\mathbb{C}^{m\times n}\times\mathbb{C}^m:\\ \|A\|_F\leq\sqrt{2},\|b\|_2\leq\sqrt{2}}} \|f(A,b)-\hat{f}(A,b)\|_2 < \infty$ *for* $f \in \{f_{lsq\text{-}m}, f_{lsq\text{-}n}\}$.

Remark 7. *Since the results are also valid in Borel-Turing sense, effective algorithmic computations on digital hardware of the problems posed by (I)-(VI) in the sense of Remark 4 are not feasible for accuracies below the introduced lower bounds. In particular, except for f_{lsq}, there does not exist a finite uniform bound: any algorithm will potentially make arbitrarily large errors on any sufficiently large compact input domain.*

How do these findings relate to intuition and empirical evidence? In particular, can we identify input instances or domains that lead to algorithmic failure? Addressing the first question, we analyze iterative methods with convergence guarantees and identify the level at which the breakdown occurs, yielding concrete insights into the second question.

B. Non-Effectiveness of Iterative Processes

It is important to stress that Theorem 6 does not contradict the existence of procedures that compute the pseudoinverse with provable convergence guarantees. For a non-zero matrix $A \in \mathbb{C}^{m\times n}$ with rank p and some $0 < \alpha < 2\|A\|_2^{-2}$, the iterative process $(A_k)_{k\in\mathbb{N}}$ following

$$A_k := \begin{cases} \alpha A^H & : \text{ if } k = 0, \\ 2A_{k-1} - A_{k-1}AA_{k-1} & : \text{ if } k \in \mathbb{N}, \end{cases} \tag{1}$$

converges to the pseudoinverse $A^\dagger$ for $k \in \mathbb{N} \cup \{0\}$ with

$$\|A^\dagger - A_k\|_2 \leq \frac{\|A\|_2}{\lambda_p(A^HA)}(1 - \alpha\lambda_p(A^HA))^{2^k}, \tag{2}$$

where $\|\cdot\|_2$ denotes the spectral norm and $\lambda_p(A^HA) > 0$ the p-th largest eigenvalue of A^HA [27]. In particular, one obtains quadratic convergence rate

$$\|A^\dagger - A_k\|_2 \leq \|A\|_2 \|A^\dagger - A_{k-1}\|_2^2 \quad \text{for } k \in \mathbb{N}. \tag{3}$$

At first glance, this suggests a uniform effective algorithm: the sequence $(A_k)_{k\in\mathbb{N}}$ converges to the pseudoinverse and is computable. Indeed, the iteration step in (1) can be expressed by the Borel-Turing computable function $f : \mathbb{C}^{n\times m} \times \mathbb{C}^{m\times n} \to \mathbb{C}^{n\times m}$ given by $f_B(X) := f(X, B) = 2X - XBX$ as

$$A_k = f_A(A_{k-1}) = f_A(f_A(\cdots f_A(A_0))) \quad \text{for } k \in \mathbb{N}. \tag{4}$$

Since each iteration step is given by a Borel–Turing computable function and computability is preserved under composition, this seemingly yields a constructive effective algorithm. However, an implementable algorithm must also start with computable initialization A_0 (otherwise the iteration can not be Borel-Turing computable) leading to convergence and, crucially, a stopping criterion that guarantees a prescribed error bound. Ideally, the algorithm would take an error parameter $N \in \mathbb{N}$ as additional input and halt if, for some $m \in \mathbb{N}$, the approximation error of A_m is smaller than 2^{-N}. Therefore, an effective implementation takes a representation of any (non-zero) matrix $A \in \mathbb{C}^{m\times n}$ and an error parameter N as input and outputs (an approximation of) $A^\dagger$ up to an error of 2^{-N}. The key observation is that the initialization and the termination of the iteration are integral parts of the effective implementation.

Formally, the question of effective implementation can be formulated as: Does there exist a Borel-Turing computable function $G : \mathbb{C}^{m\times n} \to \mathbb{C}^{n\times m}$ such that $(A_k)_{k\in\mathbb{N}}$ computed according to (4) with $A_0 = G(A)$ converges effectively to the pseudoinverse $A^\dagger$. In other words, does there exist a recursive function $e_G : \mathbb{N} \to \mathbb{N}$ such that, for any $A \in \mathbb{C}^{m\times n}$ with $G(A) = A_0$,

$$\|A^\dagger - A_k\|_2 \leq \frac{1}{2^N} \text{ holds true } \forall N \in \mathbb{N} \text{ and } k \geq e_G(N)? \tag{5}$$

If G and e_G were to exist, then the representation of any non-zero matrix $A \in \mathbb{C}^{m\times n}$ could be effectively transformed into a representation of its pseudoinverse, implying Borel–Turing computability of $f^\dagger$ on $\mathbb{C}^{m\times n} \setminus \{0\}$ and thus contradicting 6. In particular, (5) cannot even be realized for $N = 1$ on a compact input set $\|A\|_F \leq \sqrt{2}$—the proof verifies that excluding the zero matrix from the input domain does not affect this conclusion.

Corollary 8. *For $n, m \geq 2$, there does not exist a Borel-Turing computable function $G : \mathbb{C}^{m\times n} \to \mathbb{C}^{n\times m}$ to compute an initialization $A_0 = G(A)$ such that the sequence $(A_k)_{k\in\mathbb{N}}$,*

$$A_k = 2A_{k-1} - A_{k-1}AA_{k-1} \quad \text{for } k \in \mathbb{N},$$

converges effectively to the pseudoinverse $A^\dagger$.

Since the spectral norm is bounded by the (computable) Frobenius norm, fixing a computable function G, which provides a suitable initialization via (1), is straightforward. However, the associated recursive function e_G measuring the convergence rate via (2) does not exist because determining the rank of a matrix $A \in \mathbb{C}^{m\times n}$ and finding a (non-zero) lower bound on the $\mathrm{rank}(A)$-th largest eigenvalue of $A^H A$ cannot be effectively computed without further information—hereby, determining the rank effectively is the prohibitive step. On the other hand, providing additional information about the input matrices by restricting the input domain, e.g., in the edge case to contain exactly one computable element, the iterative process yields an effective approach, as the next result shows.

Proposition 9. *For any function (I)-(VI), i.e., $f \in \{f^\dagger, f^{\|\dagger\|}, \kappa, f_{lsq}, f_{lsq\text{-}m}, f_{lsq\text{-}n}\}$, and computable input $x \in dom(f)$, $f(x)$ is also computable.*

Remark 10. *Corollary 8 and Proposition 9 illustrate the contrast between effective initialization and effective convergence of the iterative process to obtain the pseudoinverse. If an effective initialization is implemented, then effective convergence is not feasible (Corollary 8). Likewise, if effective convergence is guaranteed, then the corresponding initialization cannot be performed effectively but needs to be manually crafted (Proposition 9).*

The preceding discussion shows that iterative methods necessarily rely on heuristic choices for initialization and stopping criteria. While these may perform well for certain matrix classes, they cannot guarantee correctness in the effective sense in general. In particular, the proof of Proposition 9 is non-constructive and does not extend to a uniform algorithm taking arbitrary computable inputs, as it depends on instance-specific knowledge. This raises the question of whether effective (rather than heuristic) approaches can still be obtained for restricted matrix classes. As we show next, this is indeed the case.

C. Full description of computability breakdown

Can we identify structural properties of matrices that cause the computability breakdown, and thereby delineate domains that admit effective treatment? The proof of Theorem 6 and the analysis of the SVD in Section III-A indicate that non-computability arises only in specific situations. The following result shows that it can be avoided on restricted domains, though not on sufficiently general ones.

Theorem 11. *For any function (I)-(VI), i.e., $f \in \{f^\dagger, f^{\|\dagger\|}, \kappa, f_{lsq}, f_{lsq\text{-}m}, f_{lsq\text{-}n}\}$, there exists a Turing machine that transforms a representation of any computable input $x \in dom(f)$ into a representation for $f(x)$ provided that each member of the representation of x has the same rank as x itself.*

Remark 12. *Note that Theorem 11 does not imply Borel-Turing computability. Indeed, the non-compliance with the rank property is exactly the condition exploited in the proof of the non-approximability result in Theorem 6.*

Together, Theorems 6 and 11 yield a complete characterization of the algorithmic approximability of the pseudoinverse and the functions (I)–(VI):

- If the rank property holds, the rank of any input matrix can be computed from its representation, and the pseudoinverse can then be approximated to arbitrary accuracy via the constructive procedure in Theorem 11.
- In some settings, the rank property can be enforced effectively, i.e., there exists an effective algorithm that transforms any given representation into a representation meeting the rank property. Examples include the class of full-rank matrices in $\mathbb{C}^{m\times n}$ and the set $\mathbb{C}^{m\times n}_\varepsilon := \{A \in \mathbb{C}^{m\times n} : |A_{ij}| > \varepsilon \text{ or } A_{ij} = 0\ \forall i,j\}$ for any $\varepsilon > 0$. On such domains, the pseudoinverse mapping is Borel–Turing computable. However, membership in these sets is not effectively decidable in general; at best, one can semi-decide whether an input belongs to a "benign" class of matrices (but does not effectively distinguish between benign and non-benign classes).
- Non-approximability arises when the input domain contains or is arbitrarily close to matrices of differing rank (without additional restrictions on input instances). This aligns with the discontinuities of the pseudoinverse [54], and reflects the ability to encode non-computable problems in such domains leading to a non-approximability result. In contrast, domains like the full-rank matrices or $\mathbb{C}^{m\times n}_\varepsilon$ prevent rank transitions—either by openness or by explicit constraints imposing testable conditions—thereby restoring computability. By comparison, the set of matrices with fixed rank smaller than $\min(m,n)$ also contains only matrices with equal rank. However, the set is closed so that instances on the boundary admit representations solely consisting of matrices with a different rank, which reintroduces non-computability.

Our characterization of the computability breakdown is consistent with prior results in computable linear algebra on solving linear systems [37], [38], which is natural given the relation to the pseudoinverse. This motivates a more refined analysis of the breakdown, in particular its severity relative to other non-computable problems. A suitable framework for such a comparison is Weihrauch reducibility and the associated degrees [55], which classify problems by whether they can be computed using a single oracle call to another problem. Our analysis of the failure of SVD suggests that the non-computability of the pseudoinverse is structurally simple: in particular, tracing the obstruction to zero comparisons in the SVD indicates a close connection to robust division [56], and suggests that the pseudoinverse may be Weihrauch equivalent to this problem. A formal proof of this equivalence is left for future work.

REFERENCES

[1] H. Boche, A. Fono, and G. Kutyniok, "Turing meets Moore-Penrose: Computing the pseudoinverse on Turing machines," *arXiv:2212.02940*, 2025.

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

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

[4] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Inf.*, vol. 2, p. 15023, 2016.

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[6] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation*. Graduate studies in mathematics, American Mathematical Soc., 2002.

[7] R. P. Feynman, "Quantum mechanical computers," *Optics News*, vol. 11, no. 2, pp. 11–20, 1985.

[8] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[9] J. M. Gambetta, J. M. Chow, and M. Steffen, "Building logical qubits in a superconducting quantum computing system," *npj Quantum Inf.*, vol. 3, no. 1, p. 2, 2017.

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

[11] Y. N. Böck, H. Boche, Z. G. del Toro, and F. H. P. Fitzek, "Feynman meets Turing: Computability aspects of quantum compiling revisited," *IEEE Trans. Comput.*, vol. 75, no. 2, pp. 516–526, 2026.

[12] Y. N. Böck, H. Boche, and F. H. P. Fitzek, "Feynman meets Turing: Computability aspects of exact circuit synthesis, gate efficiency, and the spectral gap conjecture," *IEEE Trans. Quantum Eng.*, vol. 7, pp. 1–31, 2026.

[13] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, 1995.

[14] H. F. Chau and F. Wilczek, "Simple realization of the Fredkin gate using a series of two-body operators," *Phys. Rev. Lett.*, vol. 75, pp. 748–750, 1995.

[15] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Phys. Rev. Lett.*, vol. 74, pp. 4091–4094, 1995.

[16] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys*, vol. 52, no. 6, pp. 1191–1249, 1997.

[17] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quant. Inf. Comput.*, vol. 6, pp. 081–095, 2006.

[18] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, 2009.

[19] S. Barz, I. Kassal, M. Ringbauer, Y. O. Lipp, B. Dakić, A. Aspuru-Guzik, and P. Walther, "A two-qubit photonic quantum processor and its application to solving systems of linear equations," *Scientific Reports*, vol. 4, 2014.

[20] X.-D. Cai, C. Weedbrook, Z.-E. Su, M.-C. Chen, M. Gu, M.-J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, "Experimental quantum computing to solve systems of linear equations," *Phys. Rev. Lett.*, vol. 110, 2013.

[21] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195–202, 2017.

[22] J. Liu, M. Liu, J.-P. Liu, Z. Ye, Y. Wang, Y. Alexeev, J. Eisert, and L. Jiang, "Towards provably efficient quantum algorithms for large-scale machine-learning models," *Nat. Commun.*, vol. 15, p. 434, 2024.

[23] P. Denning, D. Comer, D. Gries, M. Mulder, A. Tucker, A. Turner, and P. Young, "Computing as a discipline," *Computer*, vol. 22, no. 2, pp. 63–70, 1989.

[24] E. H. Moore, "On the reciprocal of the general algebraic matrix," *Bull. Am. Math. Soc.*, vol. 26, pp. 394–395, 1920.

[25] R. Penrose, "A generalized inverse for matrices," *Math. Proc. Camb. Philos. Soc.*, vol. 51, no. 3, p. 406–413, 1955.

[26] J. C. A. Barata and M. S. Hussein, "The Moore–Penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1, pp. 146–165, 2012.

[27] A. Ben-Israel and D. Cohen, "On iterative computation of generalized inverses and associated projections," *SIAM J. Numer. Anal.*, vol. 3, no. 3, pp. 410–419, 1966.

[28] S. L. Campbell and C. D. Meyer, *Generalized inverses of linear transformations*. SIAM, 2009.

[29] I. Bajo, "Computing Moore–Penrose inverses with polynomials in matrices," *Am. Math. Mon.*, vol. 128, no. 5, pp. 446–456, 2021.

[30] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*. Springer New York, NY, 2003.

[31] X. Sheng and G. Chen, "A note of computation for M-P inverse $A^{\dagger}$," *Int. J. Comput. Math.*, vol. 87, no. 10, pp. 2235–2241, 2010.

[32] J. Ji, "Gauss–Jordan elimination methods for the Moore–Penrose inverse of a matrix," *Linear Algebra Its Appl.*, vol. 437, no. 7, pp. 1835–1844, 2012.

[33] V. Pan and R. Schreiber, "An improved Newton iteration for the generalized inverse of a matrix, with applications," *SIAM J. Sci. Stat. Comput.*, vol. 12, no. 5, pp. 1109–1130, 1991.

[34] T. Söderström and G. W. Stewart, "On the numerical properties of an iterative method for computing the Moore–Penrose generalized inverse," *SIAM J. Numer. Anal.*, vol. 11, no. 1, pp. 61–74, 1974.

[35] M. Ziegler and V. Brattka, "Computing the dimension of linear subspaces," in *SOFSEM 2000: Theory and Practice of Informatics*, pp. 450–458, Springer Berlin Heidelberg, 2000.

[36] V. Brattka and M. Ziegler, *Computability of Linear Equations*, pp. 95–106. Boston, MA: Springer US, 2002.

[37] M. Ziegler and V. Brattka, "Computability in linear algebra," *Theor. Comput. Sci.*, vol. 326, no. 1, pp. 187–211, 2004.

[38] M. Ziegler, "Real computation with least discrete advice: A complexity theory of nonuniform computability with applications to effective linear algebra," *Ann. Pure Appl. Log.*, vol. 163, no. 8, pp. 1108–1139, 2012.

[39] M. Ziegler and V. Brattka, "A computable spectral theorem," in *Computability and Complexity in Analysis*, pp. 378–388, Springer Berlin Heidelberg, 2001.

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

[41] J. Myhill, "A recursive function, defined on a compact interval and having a continuous derivative that is not recursive.," *Michigan Mathematical Journal*, vol. 18, no. 2, pp. 97 – 98, 1971.

[42] M. B. Pour-El and N. Zhong, "The wave equation with computable initial data whose unique solution is nowhere computable," *Mathematical Logic Quarterly*, vol. 43, no. 4, pp. 499–509, 1997.

[43] H. Boche and V. Pohl, "Turing meets circuit theory: Not every continuous-time LTI system can be simulated on a digital computer," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 67, no. 12, pp. 5051–5064, 2020.

[44] D. Elkouss and D. Pérez-García, "Memory effects can make the transmission capability of a communication channel uncomputable," *Nat. Commun.*, vol. 9, 03 2018.

[45] R. F. Schaefer, H. Boche, and H. V. Poor, "Turing meets Shannon: On the algorithmic computability of the capacities of secure communication systems (invited paper)," in *2019 IEEE Int. Workshop Signal Process. Adv. Wirel. Commun. (SPAWC)*, pp. 1–5, 2019.

[46] H. Boche and V. Pohl, "On the algorithmic solvability of spectral factorization and applications," *IEEE Trans. Inf. Theory*, vol. 66, no. 7, pp. 4574–4592, 2020.

[47] H. Boche and U. J. Mönich, "Turing computability of Fourier transforms of bandlimited and discrete signals," *IEEE Trans. Signal Process.*, vol. 68, pp. 532–547, 2020.

[48] R. I. Soare, "Recursively enumerable sets and degrees," *Bull. Am. Math. Soc.*, vol. 84, pp. 1149–1181, 1987.

[49] K. Weihrauch, *Computable Analysis: An Introduction*. Berlin, Heidelberg: Springer-Verlag, 2000.

[50] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics*. Perspectives in Logic, Cambridge University Press, 2017.

[51] R. Downey, ed., *Turing's legacy: developments from Turing's ideas in logic*. Lecture Notes in Logic, Cambridge University Press, 2014.

[52] S. Cooper, *Computability Theory*. Chapman Hall/CRC Mathematics Series, CRC Press, 2017.

[53] K.-I. Ko, *Complexity Theory of Real Functions*. USA: Birkhauser Boston Inc., 1991.

[54] V. Rakočević, "On continuity of the Moore-Penrose and Drazin inverses," *Matematichki Vesnik*, vol. 49, no. 3-4, pp. 163–172, 1997.

[55] V. Brattka and G. Gherardi, "Weihrauch degrees, omniscience principles and weak computability," *J. Symb. Log.*, vol. 76, no. 1, pp. 143–176, 2011.

[56] T. Kihara and A. Pauly, "Dividing by zero - how bad is it, really?," in *41st Int. Symp. Math. Found. Comput. Sci.*, vol. 58 of *Leibniz International Proceedings in Informatics*, pp. 58:1–58:14, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.