Effects of Imperfections on Quantum Algorithms: A Software Engineering Perspective

Felix Greiwe
Technical University of Applied Sciences Regensburg
Regensburg, Germany
felix.greiwe@oth-regensburg.de

Tom Krüger
Technical University of Applied Sciences Regensburg
Regensburg, Germany
tom.krueger@oth-regensburg.de

Wolfgang Mauerer
Technical University of Applied Sciences Regensburg
Siemens AG, Technology
Regensburg/Munich, Germany
wolfgang.mauerer@othr.de

***Abstract*—Quantum computers promise considerable speedups over classical approaches, which has raised interest from many disciplines. Since any currently available implementations suffer from noise and imperfections, achieving concrete speedups for meaningful problem sizes remains a major challenge. Yet, imperfections and noise may remain present in quantum computing for a long while. Such limitations play no role in classical software computing, and software engineers are typically not well accustomed to considering such imperfections, albeit they substantially influence core properties of software and systems.**

In this paper, we show how to model imperfections with an approach tailored to (quantum) software engineers. We intuitively illustrate, using numerical simulations, how imperfections influence core properties of quantum algorithms on NISQ systems, and show possible options for tailoring future NISQ machines to improve system performance in a co-design approach.

Our results are obtained from a software framework that we provide in form of an easy-to-use reproduction package. It does not require computer scientists to acquire deep physical knowledge on noise, yet provide tangible and intuitively accessible means of interpreting the influence of noise on common software quality and performance indicators.

***Index Terms*—noisy quantum computing, NISQ systems, quantum software engineering, HW-SW co-design**

I. Introduction

Quantum computing promises improvements and computational speedups over classical approaches for many tasks and problems, which include cryptography [1], machine learning [2], optimisation [3], [4], or simulating chemical and physical systems [5]. This has raised considerable interest across scientific communities, including (quantum) software engineering—programmable quantum computers and appliances will, eventually, involve software in one form or another.

Given the current state of available noisy intermediate scale quantum (NISQ) hardware [6], actual quantum advantages are rarely seen (except for specially crafted problems [7], [8]). While error correction techniques for quantum computers exist, the required hardware resources exceed current system dimensions by many orders of magnitude [9]. Therefore, imperfections in quantum computers will be present in the forseeable future, and it is important for SW engineers and researchers to be aware how low level effects like noise influence software qualities. Given these conditions, evaluating, characterising and predicting functional and non-functional properties of quantum software is a complex, multi-facetted endeavour that requires catering for many details, many of them are unaccustomed from classical software engineering. It is becoming increasingly clear that possible performance benefits of quantum systems will be available only under particular circumstance that concern both, algorithms and hardware. Seemingly straightforward approaches (or naive analogies with classical systems and software) can quickly lead to bogus, unreliable or downright wrong statements that mis-characterise potential benefits of quantum approaches. Our paper is intended to help software engineers develop a realistic expectation regarding the performance of quantum algorithms under noise, on different types of hardware. We provide illustrative examples that show impacts on a number of seminal (classes of) algorithms—Grover search, quantum Fourier transform, and variational quantum circuits. Our main contributions are as follows:

- We provide a self-contained exposition on modelling noise and imperfections tailored at computer scientists and software engineers to create a tangible bridge between fundamental physics and non-functional properties conventionally employed in software engineering.
- Using numerical simulations, we show the influence of typical imperfections on multiple seminal algorithms for different hardware classes, and provide an intuitive understanding on potentials of hardware-software co-design for future quantum computing systems.
- We provide a reproduction package [10] on the supplementary website (link in PDF)[1] that allows software engineers to quickly evaluate how noise and imperfections influence their designs, without having to acquire a deeper understanding of low-level physical details.

Our contribution intends to increase awareness in the quantum software community on the impact of noise and imperfections on algorithmic performance, but also on the opportunities of co-designing future (NISQ) systems whose properties are favourable for specific classes of applications. In contrast to existing work, our paper places stronger focus on providing self-contained instructions on *how* to understand and model imperfections, and how to judge their influence on key qualities or requirements of software and software

[1]DOI-compliant version: https://doi.org/10.5281/zenodo.8001512

architectures. This is, for instance, required to support a well-informed discussion on finding proper levels of abstraction needed to decouple peculiarities of QPUs as good as possible, yet should not stand in the way of utilising the computational power provided by QPUs. Likewise, knowledge of imperfections at a reasonable level of detail can help researchers to avoid placing inflated expectations on the capabilities of quantum approaches.

The code in the reproduction package is based on the open source framework Qiskit [11], and does not depend on any real quantum hardware or proprietary compilers, which makes it accessible to a wide audience. It not only enables researchers to easily re-create our results, but has, instead, especially been designed to enable researchers and software engineers to extend it with own algorithms (and test/benchmarking cases), and study them under the influence of various types of noise and gate sets, without having to manually implement the required physics-centric evaluation mechanisms.

The remainder of this paper is structured as follows: After reviewing related work in quantum software engineering and quantum noise in Sec. II, we discuss important characteristics of current QPUs, as well as particularly relevant open hardware challenges, in Sec. III. We provide a gentle introduction to modelling noise and imperfections tailored towards software engineers in Section IV, and illustrate these considerations by discussing their impact on several seminal quantum algorithms in Sec. V, followed by a discussion of the implications for software engineering in Sec. VI. We conclude in Sec. VII.

II. Related Work

Since quantum software engineering is in its initial stages (yet, Piattini *et al.* [12] go as far as to proclaim a new "golden age" of software engineering), the available literature still is sparse, and noise and imperfections are ignored in (or deemed irrelevant for) many expositions that concentrate on possible future higher-level abstractions to quantum software engineering and quantum programming. For instance, Perez-Castillo *et al.* [13] discuss how to extend the unified modeling language to quantum circuits. Similarly, Gemeinhardt *et al.* [14] suggest model-driven quantum software engineering as an abstraction that extends established SWE methods. Differences between quantum and classical engineering in terms of bug patterns are studied by Campos and Souto [15], as well as Zhao [16]. Zhao [17] provides a detailed review of the available literature. Piattini *et al.* [18] suggest principles for the future development of quantum software engineering, and highlight hybrid algorithms and the desirable independence of specific quantum software frameworks. Leymann *et al.* [19] focus on often ignored aspects of imperfections in quantum computing. Structured approaches for benchmarking software on quantum computers are considered by Becker *et al.* [20] and Tomesh *et al.* [21]; in particular, the approach by Resch *et al.* [22] especially highlights the importance of choosing appropriate noise models. Salas *et al.* [23] consider noise effects on Grover's algorithm and state error thresholds.

The performance of NISQ-era variational quantum algorithms, particularly in the QAOA family, has been subject to intensive research; recent results include Refs. [24], [25]. Other application fields like machine learning (see, *e.g.*, Refs. [26], [27]) have received similar consideration from an algorithmic benchmarking and performance analysis point of view. Interestingly, it is known that noise need not necessarily be detrimental, but can also contribute improvements, as recent research (*e.g.*, Refs. [28], [29]) demonstrates. The physics-centric literature on quantum noise is extensive, and reaches considerably further back; the seminal exposition by Gardiner and Zoller [30] contains many of the fundamental results. Noise and imperfections in all possible implementation platforms for quantum computers from a physical point of view have likewise been considered in substantial depth and breadth, for which Bharti *et al.* [31] provide a review.

Characterising the capabilities of quantum computers is, in general, an active field of research: Considerations based on cross entropy [32] and quantum volume [33] consider properties of random circuits, and aim at a generically usable comparison metric that is applicable across implementation techniques, but does not allow for deriving concrete statements on algorithms or use-case scenarios. Application oriented benchmarks (*e.g.*, Refs. [34]–[36]), and other domain-specific (*e.g.*, [37], [38]) or generic (*e.g.*, Refs. [39]–[41]) approaches, consider more concrete perspectives, but often use techniques that are unaccustomed for software engineers. We aim, in contrast, at a correct, yet tangible and algorithm-oriented approach that is accessible and useful for the software engineering community.

III. Quantum Hardware and HW Challenges

One major challenge in quantum computing is to provide an isolation between the fragile quantum bits that carry quantum information, and are used to perform computations on, and the surrounding environment. Interactions between qubits and the environment lead to the loss of quantum information (*decoherence*), and therefore degrade the quality of computational processes. Likewise, operations on one or more of the qubits that perform the actual computation may be imperfect, and usually implement a transformation that is only "close to" the actual specification, including random variations. Both aspects do not occur in classical systems (or can be very well countered), and correspondingly, software engineers (outside, probably, highly specialised domains like safety-critical engineering) need not be concerned with the corresponding phenomena.

It is still unclear which basic physical concepts will provide the basis of future quantum computers. A multitude of possible approaches are currently developed and investigated, including systems based on trapped ions, neutral atoms, superconducting semiconductor-based implementations, or photonic systems.

We chose two common architectures of commercial interest to highlight the essential, far-reaching differences in their physical implementation that, as we will argue in this paper

based on numerical simulations, substantially impact many properties of systems that are relevant to software engineering.

A. Physical Foundations

1) Trapped Ions: By using an electromagnetic field to hold ions together in a trap, they serve as building blocks to realise qubits by using stable (internal) electronic states of the ions together with so-called (external) collective quantised motion states of all ions assembled in the trap. Laser pulses are used to control and couple the internal and motion states, realise single- and multi-qubit gates, and cool (slow down) the ions to motional lowest-energy states (see. *e.g.*, Ref. [42] for details).

A salient characteristic of the motional coupling that affects all quantum bits is that two-qubit operations can be executed between any two qubits, which means that the system provides a fully-meshed coupling graph. Likewise, very high gate fidelities and coherence times can be achieved, as summarised in Table I. Despite laser cooling of the involved ions, the overall system operates at room temperature [43].

In contrast to these advantages, gate execution times are comparatively large; operations require microseconds of processing time. Additionally, it is not straightforward to scale trapped ion systems to more than, say, 100 qubits, while maintaining the motional coupling between ions. Noise in gate application arises from variations of intensities and phases of of the lasers involved, but also from external electromagnetic fields that cannot be completely shielded off.

2) Superconducting Transmons: Superconducting quantum computing exploits quantum mechanical properties of macroscopic structures that stem from Cooper electron pairs that form at very low temperatures in superconductors.

The need for such low temperatures is a disadvantage compared to trapped ion systems. Also qubits are coupled to qubits in their direct neighbourhood, and achievable coherence times are orders of magnitudes lower. However, gate times in superconducting devices are in the nanosecond range, and larger systems (in terms of qubits count) can be built compared to trapped ion system. Finally, the manufacturing process can benefit from established industrial semiconductor know-how.

B. Challenges

Several limitations of current quantum computers extend across physical realisations and need to be considered when designing quantum software components, or when planning experiments to judge feasibility or scalability of proposed quantum architectures. Some of the limitations are specific to NISQ systems, others concern intrinsic limitations of quantum computers and algorithms that need to be taken into account for any consideration relating to quantum software engineering or architecture. Noise (*i.e.*, effects of imperfect quantum information representation and manipulation), limited connectivity between qubits, and gate timing characteristics offer substantial potential for future engineering improvements; they can be seen as hardware "parameters" to a certain extent from a SWE point of view. We study the respective potentials in detail the next section.

1) Noise: Quantum states are fragile, and operations on such states require involved physical manipulation techniques that are hard to implement perfectly—any real-world implementation slightly deviates from a theoretically desired perfect operation. Likewise, information in quantum states is perturbed by interaction with an outside environment, which is unavoidable because of the need to interact with and manipulate the states to perform computations. The (in)stability of quantum states and quantum operations is characterised by established measures that we discuss below; representative measures for three different commercially accessible platforms are shown in Table I.

The *coherence times* T_1 and T_2 indicate how resilient the information stored in qubits is against perturbations (longer times are better). T_1 gives the average time it takes a qubit to "relax" from $|1\rangle$ to state $|0\rangle$ (bit flip).

The stability of the relative phase in a superposition state $|+\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$ is quantified by T_2, providing the average time after which $|+\rangle$ has evolved into an equal-probability classical mixture of $|+\rangle$ and $|-\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$ (phase flip).

The quantities $e_1 = 1 - F_1$ and $e_2 = 1 - F_2$ describe error rates for one- and two qubit gates, and relate to the average gate fidelities $F_{1/2}$, which measures gate quality (an exact definition follows later). Similarly, TG_1 and TG_2 denote average gate times of one and two qubit gates, whereas n specifies the number of available qubits, and C is coupling density (*i.e.*, the average fraction of degree of connections between qubits; 100% for a fully meshed graph that represents physical all-to-all connectivity).

The systems characterised in Table I, albeit they only represent a fraction of the current variation in implementation technologies,[2] exhibit widely varying characteristics that are not straightforward to translate into established quality, performance, or scalability indicators, as they are typically considered in software engineering. Therefore, empirical characterisation and a generic understanding of the impact of imperfections on software qualities, as we address it in this paper, seems indispensable.

We obtained the low level metrics for IBM-Q system from so called *FakeBackends* which are embedded in Qiskit and are based on snapshots of their systems. For the special case of the Z and Rz gate, we set the respective gate time and error in our noise model to zero, since these gates are implemented virtually by all represented vendors. For circuit depth estimation, these gates are not considered either. For IonQ the low level metrics were taken from [44]. Since the data for T_1 coherence has a wide range we decided on a value

[2]It would have been desirable to include additional vendors and approaches in our simulations. Yet at the time of writing, public availability of the corresponding low-level data is scarce, and many vendors are reluctant to publish specific values. While we rely on vendor-reported error rates for the available data, it needs to be kept in mind that details of how these numbers were obtained are not always clearly specified. Since (commercial) vendors might be interested in a favourable representation of their products, any simulations based on these numbers should be used as indicators, not as absolute and scientifically verified performance measures.

Implementation Technology	Vendor	System	T_1	T_2	F_1	F_2	TG_1	TG_2	n	C
Superconducting Qubits	IBM-Q	Kolkata	$109.90\mu s$	$96.80\mu s$	99.968%	98.909%	$35.56ns$	$415.37ns$	27	7.98%
Trapped Ions	IonQ	Aria	$10s - 100s$	$1s$	99.95%	99.6%	$135\mu s$	$600\mu s$	21	100%
Superconducting Qubits	Rigetti	Aspen M3	$24.98\mu s$	$28.04\mu s$	99.614%	90.588%	$40ns$	$240ns$	80	3.35%

TABLE I: Low level hardware metrics for three commercially available QC platforms (see the text for details).

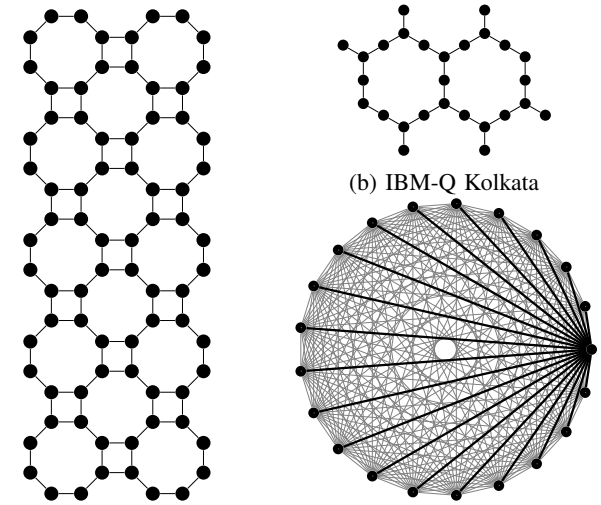
1-Qubit Gates	X	$\sqrt{X}$	R_x	R_z	GPi1	GPi2
IBM-Q Kolkata	✓	✓	✗	✓	✗	✗
IonQ Aria	✗	✗	✗	✓	✓	✓
Rigetti AspenM3	(✓)	(✓)	(✓)	✓	✗	✗
2-Qubit Gates	**C–X**	**C–Z**	**C–p**	**XY**	**MS**	
IBM-Q Kolkata	✓	✗	✗	✗	✗	
IonQ Aria	✗	✗	✗	✗	(✓)	
Rigetti AspenM3	✗	✓	✓	✓	✗	

TABLE II: Native gate sets for the investigated hardware architectures. Check marks (✓) denote supported; crosses (✗) denote unsupported gates. The parenthesised gates emphasise differences in the matrix formulation to more commonly used definitions. While we cannot discuss peculiarities of gates specific to different architectures, the reproduction package supports comparative experiments based on all gate sets.

of $50s$ in our simulations for IonQ. Vendor Rigetti provides error data obtained via randomised benchmarking as an online resource [45]. IonQ and Rigetti, to the best of our knowledge, do not state exactly whether their Rz gates are considered in calculation of average error rates for one qubit gates. Thus, the average error rates might be slight overestimations when compared to IBM-Q. This is negligible since the two qubit gates introduce errors with higher impact.

Table I shows error rates for quantum gates on different contemporary hardware approaches. All architectures are affected by noise, which limits the achievable depth of quantum circuits, and thus the computational power. Yet, there is no common noise pattern across systems, which makes most statements about performance and behaviour of quantum algorithms impossible without taking very specific hardware details into consideration, in stark contrast to classical machines. As mentioned in section I, for quantum error correction to work the requirement for thousands of qubits [46] arises which will most likely not be possible on a larger scale in the near future.

2) Connectivity: Logical quantum algorithms usually make arbitrary (pairs of) qubits interact when multi-qubit operations are applied. Most physical implementations of quantum systems place restrictions on the possible interactions between pairs of qubits. One important step in translating a logical into a physical quantum circuit is to (a) map interacting logical qubits to interconnected physical qubits, and (b) ensure that operations performed on unconnected qubits (if placement alone cannot guarantee this) are enabled by "moving" qubits into proximity prior to gate execution. While qubits cannot (for most implementation technologies) be moved physically, applying SWAP gates allows us to change to logical state between two connected qubits. This allows, at the expense of increasing circuit depths, to bring two unconnected qubits into connected positions, and then apply a joint gate operation. For architectures that do also not natively support logical swap gates, a replacement can be provided by three C–X gates, at the expense of an even larger increase in circuit depth.



(a) Rigetti Aspen M3 (b) IBM-Q Kolkata (c) IonQ Aria

Fig. 1: Hardware connectivity graphs from various commercial vendors considered in the numerical simulations. The implementations of vendors Rigetti and IBM-Q both have a grid like structure, and differ slightly in their connectivity structure. The IonQ architecture (and other trapped ion systems) features full connectivity between qubits.

Fig. 1 compares topologies for some of the major available quantum architectures. We illustrate their influence on algorithms in Section V-C.

3) Gate Sets: The set of elementary quantum gates varies considerably with implementation technology. Universal quantum computation can be achieved with many different choices. While the theoretical capabilities of each set are identical, the practical behaviour of gates under the influence of noise may vary distinctly. Executing identical algorithms on different hardware does therefore not only influence computation times (as is familiar from classical computing), but is also affected by different influence of noise. Table II illustrates elementary gate sets for the subject architectures.

C. Subject Algorithms

We have chosen three canonical, yet substantially different algorithms to study the impact of noise and imperfections: Grover search, quantum Fourier transform, and variational quantum circuits. It is possible to prove speedups over their

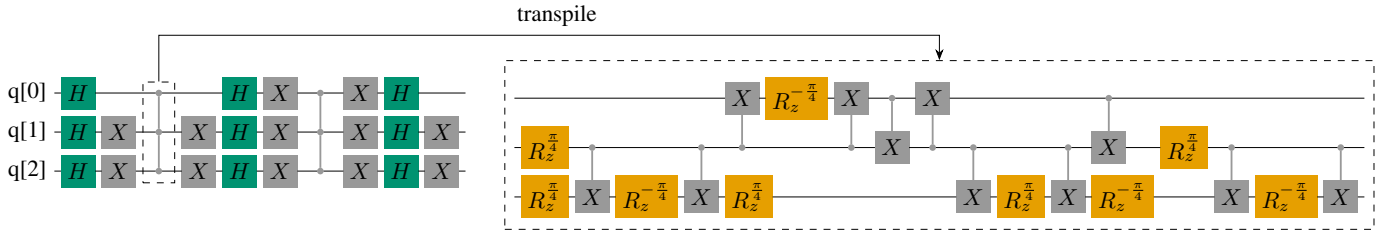


Fig. 2: Effect of transpiling a gate in a logical Grover circuit (left) to IBM-Q Kolkata hardware (right, dotted frame). Even a seemingly small component like a (multi-) controlled Z (C–Z) gate can introduce significant increase in circuit depth.

classical counterparts for the first two algorithms, albeit these only materialise for perfect, error-corrected quantum systems. Their scalability in terms of circuit depth growth with increasing input size is distinctly different.

The third class, variational quantum circuits, is speculated to exhibit speedups over classical approaches under some credible assumptions, and are particularly well suited for NISQ hardware and empirical experiments, as they allow for very shallow circuits. Yet, practically relevant speedups have still failed to materialise on a wider front in current systems.

1) Grover Search: Grover's algorithms allows for finding specific elements in an unstructured search space. Simply put, the algorithm iteratively repeats two sub-circuits: An oracle to mark the desired element in a search space, and a rotation in a two dimensional plane. For inputs of n qubits, the required number of iterations scales with $\mathcal{O}(\sqrt{2^n})$ [47], which provides a quadratic speedup compared to the best classical search algorithms—yet, this speedup is relative to exponential growth. Since the algorithm matches a wide class of application problems, it can seem tantalizing to seek "free" quantum improvements by deploying the algorithm as drop-in search replacement in existing scenarios. However, there are some pitfalls to consider: Grover search does usually not, despite common perception, query an actual "physical" database encoded in quantum states,[3] but evaluates an efficiently computable function f that acts as predicate to identify one or more optimal elements in a search space. This implies costs (especially in terms of circuit depth) to *implement* this target function using quantum operators, which may be non-trivial [52].

Many practical applications are either interested in average case complexity, or enjoy some structure in their search space that can be (also heuristically) used to speed up processing, which places considerable limitations on practical utility. Replacing classical primitives with Grover search is a commonly used pattern (*e.g.* in database research [53], [54]) in efforts to utilise quantum computing, and is sometimes backed up by evaluations on small-scale NISQ machine. We study the limited utility of this approach in Sec. V.

By determining the probability of measuring the desired element of the search space as output, we can associate a success probability with runs of Grover search. Our implementation is inspired by [52], [55].

2) Quantum Fourier Transform: The quantum Fourier transform is as a building block for many quantum algorithms, most notably Shor's factoring algorithm. It is a computational analogue of the (discrete) classical Fourier transformation, albeit there are pronounced differences in obtaining the results, as the probability of reading a Fourier coefficient is related to its magnitude, and applications that require access to the full transformation do not benefit from quantum advantage. Yet, QFT requires exponentially fewer operations than FFT.

3) Variational Quantum Circuits: The algorithmic family of variational quantum algorithms comprises circuits that contain gates whose properties are controlled by a tunable parameter. After feeding an input state through a circuit and measuring the output, the parameter settings are adjusted in a training process—not unlike classical machine learning—, and the process is repeated, until some desired target function is approximated with sufficient quality. The approach is versatile and well suited for experimentation on NISQ machines, particularly because of controllable circuit depth.

An example for a variational circuit (as we employ it in the below experiments) is shown in Fig. 5. For data encoding, which comprises the left part of our circuit, we follow the example of Ref. [56], where the given encoding proves to be well suited for function approximation. The parameterized part of our quantum circuit is one of several building blocks for variational quantum circuits, the authors of [57] investigate in their work. While the variational part of our circuit could be repeated several times, we stick to one single parameterized layer in our experiments.

For our simulations we train the variational circuit to mimic the behaviour of the function $f(x) = x^2$, extending an example of [56] for different noise levels. We will now quickly summarize the procedure: We call the parameters of the circuit $\theta_i \in \boldsymbol{\theta}$, all of which start at $\theta_i = 0$ as initial parameter values. For simplicity we choose to measure only the qubit $|q_1\rangle$ in the Z basis and subsequently extract the expectation value

[3]While it would be possible *in principle* to apply Grover search on top of quantum random access memory (QRAM), this would result in a quadratic speedup for a search task on an exponentially growing search space, which is usually irrelevant industrial settings. Other data loading alternatives exhibit similar difficulties. Approximate encoding techniques [48], together with shallow variants of Grover [49], [50], or improvements in the amplitude amplification process [51] might lead to fruition, but underline that judging non-functional software characteristics is impossible without accounting for technical and physical details that can be ignored in classical approaches. Given that QRAM is invariably harder to manufacture than classical RAM, a scenario where the former can fully replace the latter seems hardly credible.

of the observable $\langle M_j \rangle_{\boldsymbol{\theta}} \in [-1, 1]$ for the training input x_j and the current parameters. For sample based approaches the approximation of $\langle M_j \rangle_{\boldsymbol{\theta}}$ requires a reasonable choice of circuit estimations, that is, the sample number. Based on the quadratic loss function for iteration k,

$$L\left(x_j, \boldsymbol{\theta}^{(k)}\right) = \frac{1}{2}\left(\langle M_j \rangle_{\boldsymbol{\theta}^{(k)}} - f(x_j)\right)^2, \tag{1}$$

we update $\boldsymbol{\theta}$ in every iteration using gradient descent

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \frac{\partial L}{\partial \boldsymbol{\theta}}. \tag{2}$$

We calculate the derivative of the loss function using the parameter shift rule [56]

$$\begin{aligned} &\frac{\partial L}{\partial \theta_i} = \\ &\frac{1}{2}(\langle M_j \rangle_{\boldsymbol{\theta}^{(k)}} - f(x_j)) \left(\langle M_j \rangle_{\boldsymbol{\theta}^{(k)} + \vec{e}_i \frac{\pi}{2}} - \langle M \rangle_{\boldsymbol{\theta}^{(k)} - \vec{e}_i \frac{\pi}{2}}\right), \end{aligned} \tag{3}$$

where $\vec{e}_i$ is the a unit vector for component i. After carrying out the partial gradient calculation for every parameter, the accumulated gradient $\frac{\partial L}{\partial \boldsymbol{\theta}}$ is used in the parameter update.

Our simulations fix the number of training iterations at 100, and take uniformly spaced samples $x_j \in [-1, 1]$, which are permuted with a (fixed) random seed. Results are therefore comparable between architectures and noise variants.

IV. Modelling Noise and Imperfections

In the following, we outline the theoretical concepts necessary to describe and understand how noise (*i.e.*, the influence of uncontrollable external factors) impacts quantum calculations. We aim at an exposition that is accessible to software engineers without deeper involvement in quantum physics, yet sufficiently accurate to paint a realistic picture that allows for drawing reliable conclusions.

A. Mixed States and Density Operators

The notion of quantum states in software engineering usually refers to *pure states* $|\phi\rangle \in \mathbb{C}^{2^n}$, that is, states represented by a unit vector in a 2^n dimensional complex vector (Hilbert) space. Sometimes, however, it is not possible to obtain full knowledge of the state of a system. Consider, for example, the case where we take an (educated) guess whether some external influence that is not under our control (in other words, noise) flipped qubit x_i or not. In this scenario, the system is in state $|x_i\rangle$ with probability p or in state $|\neg x_i\rangle$ with probability $1-p$. Generalising this concept delivers a probability distribution $\left(\begin{smallmatrix} |\phi_1\rangle & |\phi_2\rangle & \cdots \\ p_1 & p_2 & \cdots \end{smallmatrix}\right)$ of different possible system states. If a system follows such a distribution it is said to be in a *mixed state*.

Mixed states contain *two* probabilistic aspects: (a) Stochastic outcomes resulting from measurements have their origin in the very properties of quantum theory. (b) Purely classical uncertainty about the state that arises from a *lack of knowledge* of external confounding factors (noise).

When influences beyond our active control modify a quantum state, we need to express the (classical) uncertainty arising from the scenario with classical probabilities p_i. One convenient way to express mixed states is the density matrix formalism [47], which describes a "collection" of quantum states $|\phi_i\rangle$ mixed up with classical probabilities p_i as

$$\varrho = \sum_i p_i \left|\phi_i\right\rangle \left\langle\phi_i\right|, \tag{4}$$

where $\langle\phi|$ denotes, for finite-dimensional systems, the conjugate transpose of $|\phi\rangle$, turning the object $|\phi\rangle\langle\phi|$ into a matrix. Application of a unitary operator U, which represents a computational step, to a mixed state ϱ is described by $\varrho \to U\varrho U^\dagger$. In the following, transformations of the density matrix will be used to describe evolution of quantum states suspect to noise.

B. Dynamics of Noisy Quantum Programs

The evolution of a closed quantum system is described by unitary operations. A noisy system on the contrary is open to an external environment (source of noise). The trick to model open noisy systems is to include the environment, such that one ends up with a bigger but closed quantum system. Let ϱ be the state of an open quantum system of interest, which we will call the principal system. Now, we combine ϱ with the state of its environment ϱ_{env}. The new system $\varrho \otimes \varrho_{\text{env}}$ is closed and its evolution $U(\varrho \otimes \varrho_{\text{env}})U^\dagger$ can be described by a unitary operator U. Tracing out the environment reveals the evolution of ϱ under noise: $\mathcal{E}(\varrho) = \text{tr}_{\text{env}}(U(\varrho \otimes \varrho_{\text{env}})U^\dagger)$. The partial trace is a tool in the density matrix formalism to discard certain parts of a quantum mechanical systems, for more information we refer to [47]. Note that the *quantum operator* $\mathcal{E}(\varrho)$ is not necessarily unitary anymore. Let $\mathcal{B}_e = \{|e_k\rangle\}_k$ be a basis of the environment. Now, if the environment is measured in $\mathcal{B}_e$ after the time evolution, then the outcome determines the state of the principal system. We end up with a random distribution of states for the principal system depending on the measurement. The effect the environment had on ϱ when the outcome k occurred can be described by an operator E_k [47], [58], leading to a mixed state description

$$\varrho \mapsto \sum_k E_k \varrho E_k^\dagger, \quad \sum_k E_k^\dagger E_k = \mathbb{1} \tag{5}$$

C. Fidelity

In this work we mainly focus on the effect of noise when performing computations, that is gate errors. Consider a quantum operation $\mathcal{E}(\varrho)$ describing the noise impact on ϱ. Under the influence of noise, a pure state $\varrho = |\psi\rangle\langle\psi|$ evolves to $\mathcal{E}(\varrho) = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. We can now calculate $\langle\psi|\mathcal{E}(\varrho)|\psi\rangle = \sum_i p_i \langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle = \sum_i p_i |\langle\psi|\psi_i\rangle|^2$, which measures the overlap between $|\psi\rangle$ and $\mathcal{E}(\varrho)$. This can be seen as a measure of how much information is preserved under noise. In a perfect noiseless environment $\mathcal{E}(\varrho) = |\psi\rangle\langle\psi| = \varrho$, preserving all the information, respectively $\langle\psi|\mathcal{E}(\varrho)|\psi\rangle = |\langle\psi|\psi\rangle|^2 = 1$. On the other hand, the more $\mathcal{E}(\varrho)$ turns $|\psi\rangle$ in the direction of an orthogonal state $\left|\psi^\perp\right\rangle$ the more information gets lost and $\langle\psi|\mathcal{E}(\varrho)|\psi\rangle$ approaches 0, as $\mathcal{E}(\varrho)$ goes to $\left|\psi^\perp\right\rangle\!\left\langle\psi^\perp\right|$.

The measure $F = \langle\psi|\mathcal{E}(\varrho)|\psi\rangle$ is commonly denoted as the *fidelity*;[4] it is useful to judge how much the result of a noisy quantum computer deviates form the result of a perfect machine.

The deviation of imperfect quantum gates (or other components) from a perfect implementation is not directly captured by fidelity: Gates do not operate on a fixed input state. Their degree of deviation from a perfect gate is state-dependent, so determining gate fidelity for a single state is insufficient to characterise quality. Instead, the *average fidelity* describes the mean over individual gate fidelities for all quantum states.[5] The values shown in Table I represent average fidelities.

D. Noise Models

Having introduced the general modelling principles for noisy quantum systems, we can now describe specific types of noise that represent typical physical imperfections.

1) Bit Flips: A probabilistic qubit flip [47] is given by

$$\varrho \mapsto (1-p)\mathbb{1}\varrho\mathbb{1}^\dagger + pX\varrho X^\dagger. \tag{6}$$

The operation applies the Pauli X gate (bit flip) to the one qubit system ϱ with probability p, and else leaves the state as is. Note how the above equation is one way to choose the operators E_k in (5). Similarly, the phase flip error (Pauli Z gate) and the bit-phase flip error (Pauli Y gate) can be constructed by replacing X by Z or Y in Eq. (6).

2) Depolarisation: One commonly used error in simulation is the completely depolarising operator on one qubit which randomly applies the Pauli operators X, Y, Z [47], [61]

$$\begin{aligned}\varrho \;\mapsto\; & (1-p)\mathbb{1}\varrho\mathbb{1}^\dagger + \\ & p\frac{1}{4}\left(\mathbb{1}\varrho\mathbb{1}^\dagger + X\varrho X^\dagger + Y\varrho Y^\dagger + Z\varrho Z^\dagger\right),\end{aligned} \tag{7}$$

with a certain probability, and else leaves the qubit as is. A quick calculation reveals that (7) equals $\varrho \mapsto (1-p)\varrho + p\frac{1}{2}\mathbb{1}$, where $\frac{1}{2}\mathbb{1}$ is the density representing the state of a system being in every basis state with equal probability. Hence, the system either stays intact or all information gets destroyed with probability p. For a n qubit system we get [47]:

$$\varrho \mapsto (1-p)\varrho + p\frac{1}{2^n}\mathbb{1} \tag{8}$$

3) Thermal Relaxation: The thermal-relaxation error models the decoherence of quantum system over time. The derivation of the associated quantum operator is significantly less straight forward, so we refer to Refs. [62], [63] for a derivation.

4) Hardware-Matched Composite Noise: We close by incorporating hardware metrics into a noise model (limited to stochastic gate noise, and ignoring measurements, state preparation, idle noise and coherent error models). We construct three noise models from the data in Table I.[6]

[4]Different definitions of fidelity are given in the literature; as the general characteristics are identical, we only consider one variant in this paper..

[5]How to compute the average of a desired quantity over all possible quantum states of a system is an interesting problem on its own that we cannot discuss in detail; see, for instance, Ref. [59], [60] for more information.

[6]Our approach is a simplified version of the integrated error models for IBM-Q "FakeBackends" provided (and partly explained) in Qiskit [11].

The idea is to use a composite error consisting of thermal relaxation depolarisation for every gate, such that average gate fidelities of the real hardware match the model [64]. For our simplified version we only distinguish between one and two qubit gates, but do not introduce per-gate errors, or errors depending on individual qubit quality. While this renders the model less accurate than, for instance, IBM-Q "FakeBackends", it allows us to apply it to hardware for which detailed quality data are not publicly available. The construction goes as follows: We define an error operator $\mathcal{E} = \mathcal{E}_D \circ \mathcal{E}_R$, combining both the depolarising error $\mathcal{E}_D$ (7) and the thermal relaxation error $\mathcal{E}_R$. The fidelity of $\mathcal{E}_D$ is given by [65]

$$F_D = 1 - p(1 - 2^{-n}), \tag{9}$$

and the fidelity for thermal relaxation channel $\mathcal{E}_R$ can be calculated using its parameters, namely T_1, T_2 and the gate time T_G. We tune our model to match target fidelities F_{targ} found in the literature. Using (8), the composition of depolarizing and thermal relaxation channel is $\mathcal{E}_D \circ \mathcal{E}_R = (1-p)\mathcal{E}_R(\varrho) + p\frac{1}{2^n}\mathbb{1}$. The fidelity denotes the overlap with a pure reference state $|\psi\rangle$,

$$\begin{aligned}F(\mathcal{E}_D \circ \mathcal{E}_R) =\; & \langle\psi|(1-p)\mathcal{E}_R(\varrho) + p\frac{\mathbb{1}}{2^n}|\psi\rangle \\ = \; & (1-p)\,\langle\psi|\mathcal{E}_R(\varrho)|\psi\rangle + p\,\langle\psi|\frac{\mathbb{1}}{2^n}|\psi\rangle\,,\end{aligned} \tag{10}$$

where we use the linearity of the inner product in the second argument. The left hand side of Eq. (10) is $(1-p)F(\mathcal{E}_R)$, while the right hand side is $pF(\mathcal{E}_D|_{p=1}) = p2^{-n}$, which corresponds to the fidelity of the depolarising channel evaluated at $p = 1$. We can express the target fidelity dependent on p, and solve

$$\begin{aligned}& F_{\text{arg}} = F(\mathcal{E}_D \circ \mathcal{E}_R) = (1-p)F_R + p2^{-n} \\ \Leftrightarrow\; & p = \frac{F_R - F_{targ}}{F_R - 2^{-n}}.\end{aligned} \tag{11}$$

This ensures the composition matches the vendor target fidelity F_{targ} given in Table I.

E. Measurement and Sampling

It is textbook knowledge that measuring quantum states results in probabilistic outcomes; a natural question that needs to be addressed to characterise algorithmic performance is how many samples are required to achieve acceptable trust in outcomes. Given an error margin ϵ that we are willing to accept, and a desired confidence δ for the sampled probabilities to fall within this margin of error, a lower bound on the required number of samples s can be determined by invoking the variant $s \geq \frac{1}{2\epsilon^2}\log(2/\delta)$ of the Höffding inequality (see, *e.g.*, [66]), to arrive at meaningful statements. Note that the numerical experiments considered in Sec. V simulate the complete density matrix, from which *exact* measurement statistics can be deterministically extracted. Therefore, no sampling noise as it would arise for real hardware is contained in the plots. Accessing the density matrix on real hardware is possible, but requires an (experimentally involved) quantum state tomography that measures a complete set of observables whose expectation values determine the density operator (a number of

measurements exponential in the system size is required, albeit less costly approximations are possible [67]). Consequently, sampling is unavoidable to characterise quantum algorithms on real systems, and the above considerations guide software engineers on what temporal overheads to expect.

V. Numerical Simulations

We now commence with illustrating the concrete effects of the various modes of imperfection on the subject algorithms, and show how they crucially affect many algorithmic properties that are directly relevant for software engineering. We deliberately base our considerations on two seminal, very well understood algorithms despite their known non-usefulness on NISQ hardware, as it allows us to focus on the influence of noise instead of having to consider peculiarities of the subject algorithms. This is important to approach the topic from a tangible, concrete software engineering perspective that links the influence of noise with what computer scientists are well acquainted with: The performance analysis of algorithms.

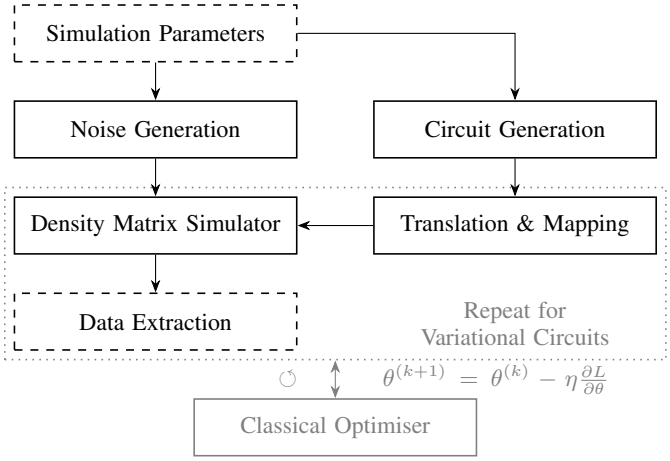


Fig. 3: Simulation procedure. Gray parts only apply to variational circuits with iterative parameter optimisation.

Fig. 3 summarises the simulation procedure as implemented in the reproduction package (available on the supplementary website). First, a quantum circuit is generated, together with a noise model characterised by type of noise and strength, as specified by the input parameters. Then, the logical circuit is translated into a physical representation for one of the supported hardware platforms, which comprises a user-customisable topology and gate set, and does not necessarily correspond to a real physical platform, albeit we focus on the platforms characterised in Table I. This allows users to consider tailored designs that match the requirements of problems of interest, and can guide (co-)development of quantum hardware. The reproduction package extends standard means provided by Qiskit with gate sets for IonQ and Rigetti, as well as methods for mapping base circuits onto these.[7]

[7]Our results are obtained from the Qiskit density matrix backend, which provides complete and accurate results, and is not restricted with respect to simulating noise. We use a standard gradient descent optimiser to iteratively improve parameters for variational algorithms.

A. Noise and Scalability

We start by considering scalability with respect to input size and noise strength. While it is well known that Grover Search and QFT can not be realistically deployed on NISQ machines, it is instructive to see how detrimental effects of noise on their performance are. Fig. 4 simulates success probabilities using the inherent native machine noise for both algorithms.

For all hardware architectures, the success probability quickly drops; Grover search does not produce valid solutions for more than six qubits on any architecture. This clearly indicates that experimental evaluations that are intended to show the practical functionality Grover-based approaches carry little merit; the principal functioning of Grover's algorithm is very well understood, but it is hardly possible to extrapolate any meaningful statements from such measurements. It is also instructive to consider how the slight differences in fidelity (recall Table II) lead to comparatively large differences in success probabilities for the different vendor platforms.

As Fig. 4 shows, success probability and state fidelity are essentially identical. While it is possible to define a "successful" target state in the computational basis for Grover and QFT, this does not necessarily hold for other approaches, and does also not extend to approximation cases where closeness to an ideal state is sought. Fidelity (as a continuous measure of closeness) is a useful replacement for success probability.

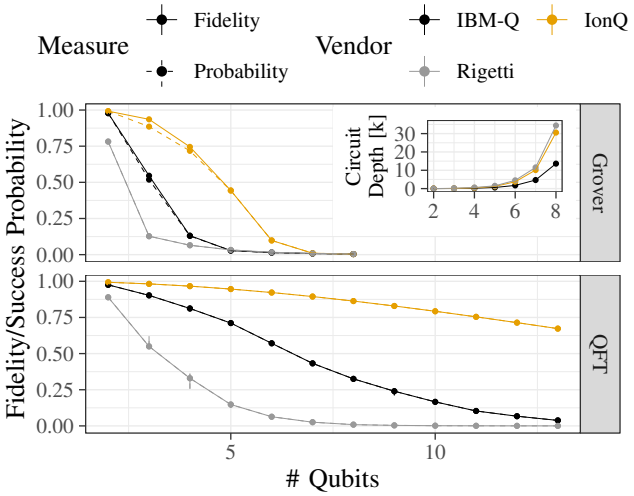


Fig. 4: Success probability and fidelity (relative to a noise-free perfect execution) for Grover Search and QFT.

The inset in the top part of Fig. 4 illustrates the increase in circuit depth with increasing input size; Fig. 7 further below provides the same information for QFT (for now, only consider the elements for native connectivity). Circuit depths for Grover increase exponentially, and exceed values of 1000 for more than 6 qubits, resulting in zero success probability. The increase for QFT is more relaxed, with depths of around 150 for all architectures at 11 qubits of input. The differences in success probability highlight circuit depth as key

performance (and feasibility) indicator, which should therefore be at the core interest of software engineers.

B. Noise Variants

The previous examples are based on noise (inspired by the characteristics of real hardware), which combines effects of different physical processes, as we have outlined above. We now evaluate individual contributions of elementary noise types that may provide guidance in designing future quantum systems. We apply these to a shallow variational quantum circuit, which is shown in Fig. 5. Parameters are trained using the procedure described in Sec. III-C3.[8]

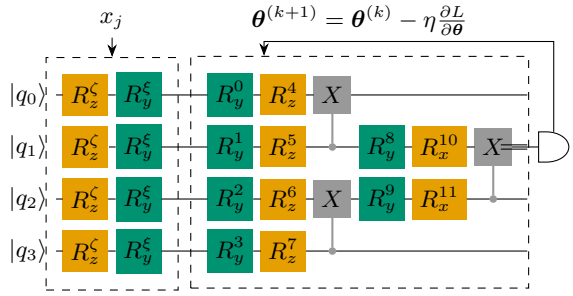


Fig. 5: Variational quantum circuit. The left part encodes input data, while the right part holds operators with trainable parameters. We abbreviate $\zeta := \operatorname{acos} x^2$ and $\xi := \operatorname{asin} x$, and set $R_\sigma^k := R_\sigma(\theta_k)$. The quantum register is set to $|q\rangle = |0000\rangle$

The training loss for Pauli-X, Y, and Z, as well as a depolarising channel, is shown in Fig. 6 for different levels of noise over 100 training iterations. The loss converges to zero once parameters have been satisfactorily learned; the simulations show that increasing noise strength impedes this process differently depending on the noise type. While bit flips (Pauli-X) are particularly obstructive, platform like trapped ion systems are particularly resilient against this type of noise [68], [69], which might be a relevant criterion for choosing an underlying quantum platform for a software system.

The right hand side of Fig. 6 illustrates how predictions obtained from the trained models degrade with increasing amounts of noise, providing tangible means of judging the effects of different types and strengths of noise. Both influence result quality in different ways, and software engineers will need to decide (by choice of hardware) which variants are best tolerable for particular use-cases considering domain knowledge and requirements.

C. Connectivity and Gate-Sets

Finally, let us elaborate on effects of connectivity structure and gate sets under the influence of noise. For given architectures, the combination is fixed, but co-designing systems that optimise either for specific algorithms is seen as one possible path towards quantum advantage [70]–[73]. Software engineers should be aware of possible future design opportunities.

The top part of Fig. 7 fixes the gate sets for each vendor, but shows circuit depth scaling for both, the native connectivity structure and a fully connected architecture (hypothetical for IBM-Q and Rigetti, standard for IonQ). Owing to the need to insert swap gates for IBM-Q and Rigetti, depth grows super-linearly with native connectivity, but increases linearly with a full mesh, substantially reducing circuit depth.[9]

For IonQ, circuit depth increases quicker than for the other architectures. This can be attributed (based on manual inspection of the generated circuits) to weaknesses of the circuit translator that maps logical to physical circuits,[10] which stresses the importance for software engineers to place greater emphasis on low-level details like compiler performance that is only of marginal interest for many classical SE tasks.

The bottom part of Fig. 7 illustrates some additional effects: A fully coupled connectivity structure combined with fixed per-gate error rates for all vendors isolates the effects of vendor specific base gates, and in particular, of compiling to them. Here, the performance of IonQ base gates is mostly due to our sub-optimal transpiler, as the resulting larger circuit depth gives more opportunity to "pick up noise". Software engineers must be aware of a possibly complex interplay of factors when evaluating algorithmic quantum performance.

VI. Implications for Software Engineering

We have illustrated how noise and imperfections impact NISQ performance, and that a certain amount of knowledge of the underlying mechanisms is necessary for proper interpretation. From the (quantum) software engineering point of view, imperfections influence if and how non-functional requirements can be satisfied. In particular, they affect scalability, performance, testability, and cost. The relation to the first two qualities has already been intensively discussed above.

Testing outcomes of quantum algorithms needs to deal with two aspects of uncertainty: Measurements leading to stochastic outcome distributions, and imperfections in gates and components. While probabilistic behaviour is well established in classical computing [75], physical imperfections have found little consideration in SW testing to the best of our knowledge. To appropriately design noise-aware tests and judge test results, software engineers need to be able to understand quantum noise at a sufficient level of detail.

The impact of noise and imperfections on quantum software could be ignored if perfect, large-scale QPUs with error correction were available. No physical reasons prevent designing the required systems, but many engineering challenges make specifying concrete roadmaps towards this goal challenging. Even if such systems can eventually be built,

[8]Note that the choice of hyper-parameters (*e.g.*, step size), as well as the classical optimiser influence algorithmic performance. While different hyper-parameters might be beneficial for different noise levels or noise methods, we did not consider such an optimisation in the scope of this paper.

[9]Manufacturing a fully meshed connectivity graph is extremely challenging for semiconductor-based approaches. However, as Refs. [70], [74] show, even small additions to existing connectivity structures can result in major improvements in circuit depth.

[10]Native IonQ compilers might improve results, yet do not satisfy our goal of providing open and reproducible means of obtaining simulation results.

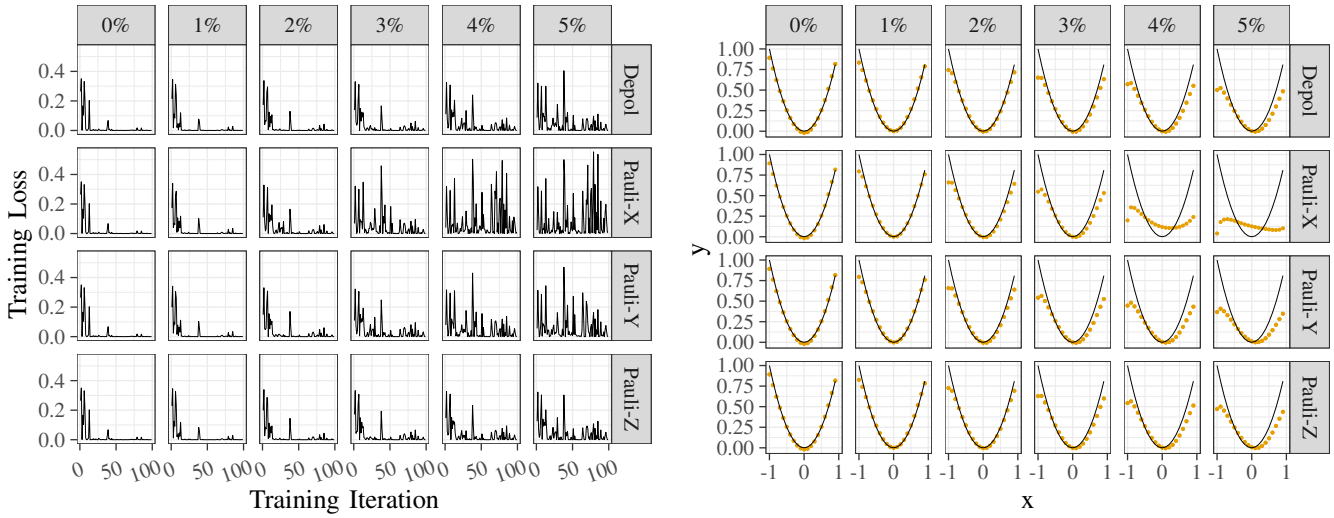


Fig. 6: Approximation of $f(x) = x^2$ using the variational quantum circuit shown in Fig. Fig. 5. (lhs) Comparison noise (applied in training and inference) variants effects on training loss. (rhs) Predictions (orange dots) versus target function (black line).

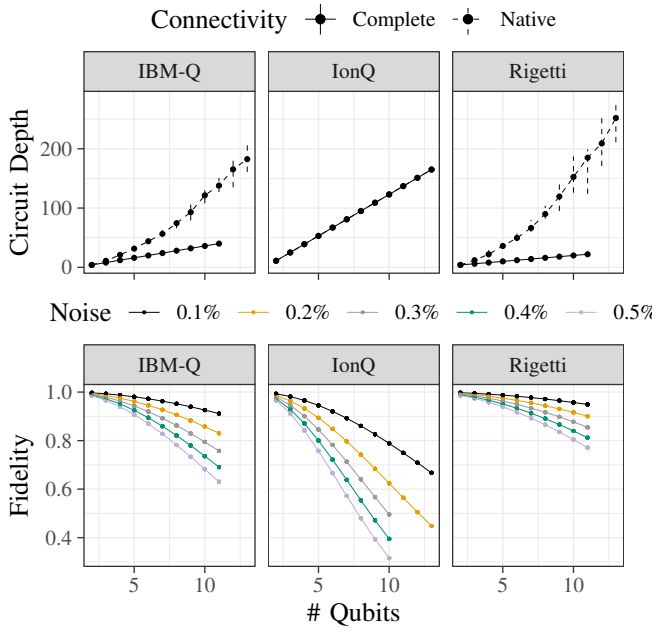


Fig. 7: Impact of vendor-specific gate sets for QFT on (a) fully connected qubits versus vendor topologies (top), and (b) on result fidelity for varying levels of depolarising noise and full connectivity (bottom). Circuit depths vary because of the stochastic transpilation.

practical industrial applications will not just be judged by performance considerations, but by their overall cost-benefit trade-off, which is among core concern of any engineering discipline. Consequently, since imperfect error correction and error mitigation schemes [76] will likely result in less costly machines, it seems reasonable to assume that such machines will co-exist with perfect quantum computers, given they can solve certain tasks advantageously over classical computers. For instance, Liu *et al.* [77] prove exponential speedups for certain types of quantum machine learning on fault-tolerant machines, which are believed to be extensible to NISQ machines using error-mitigation (Hubregtsen *et al.* [26] study training embedding kernels on NISQ machines).

Some properties of quantum states and circuits require explicit consideration in designing new and extending existing test methodology: Not just the stochastic nature of quantum measurements, but also the impact of imperfections makes defining desirable test results hard, as it is necessary to distinguish these measurement-induced variations from variations induced by noise and imperfections. Guidelines that eliminate the need for individual software engineers to be aware of statistical peculiarities could be established.

VII. Conclusion

Using a reproducible and extensible empirical simulation approach, we illustrated how noise and imperfections affect the the properties of quantum algorithms on existing and hypothetical NISQ hardware. A solid understanding of such effects is useful not only for researchers and engineers working on hardware implementations, but also for software engineers. Yet, it seems unreasonable to equip every software engineer or SWE researcher with detailed physical knowledge on noise. We instead provide a suitably detailed introduction to the topic, accompanied by an easy-to-use replication package that allows software engineers to explore the influence of noise with little effort. We deem this a crucial aspect in the endeavour of realising future quantum applications of practical benefits.

Acknowledgements

This work is supported by the German Federal Ministry of Education and Research within the funding program *Quantentechnologien – von den Grundlagen zum Markt*, contract number 13NI6092.

References

[1] A. K. Ekert, "Quantum cryptography and bell's theorem," in *Quantum Measurements in Optics*. Springer, 1992, pp. 413–418. [Online]. Available: https://doi.org/10.1103/PhysRevLett.67.661

[2] J. Biamonte, P. Wittek *et al.*, "Quantum machine learning," *Nature*, vol. 549, no. 7671, 2017. [Online]. Available: https://doi.org/10.1038/nature23474

[3] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1411.4028

[4] A. Bayerstadler, G. Becquin *et al.*, "Industry quantum computing applications," *EPJ Quantum Technology*, vol. 8, no. 1, 11 2021. [Online]. Available: https://epjquantumtechnology.springeropen.com/track/pdf/10.1140/epjqt/s40507-021-00114-x.pdf

[5] E. Altman, K. R. Brown *et al.*, "Quantum simulators: Architectures and opportunities," *PRX Quantum*, vol. 2, Feb 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.2.017003

[6] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, aug 2018. [Online]. Available: https://doi.org/10.22331%2Fq-2018-08-06-79

[7] F. Arute, K. Arya *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[8] H.-S. Zhong, H. Wang *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abe8770

[9] J. Roffe, "Quantum error correction: an introductory guide," *Contemporary Physics*, vol. 60, no. 3, pp. 226–245, 2019. [Online]. Available: https://doi.org/10.1080/00107514.2019.1667078

[10] W. Mauerer and S. Scherzinger, "1-2-3 reproducibility for quantum software experiments," *Q-SANER@IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2022.

[11] A. tA v, M. S. ANIS *et al.*, "Qiskit: An open-source framework for quantum computing," 2021.

[12] M. Piattini, G. Peterssen, and R. Pérez-Castillo, "Quantum computing: A new software engineering golden age," *ACM SIGSOFT Software Engineering Notes*, vol. 45, no. 3, pp. 12–14, 2021. [Online]. Available: https://doi.org/10.1145/3402127.3402131

[13] R. Pérez-Castillo, L. Jiménez-Navajas, and M. Piattini, "Modelling quantum circuits with uml," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, 2021, pp. 7–12. [Online]. Available: https://doi.org/10.1109/Q-SE52541.2021.00009

[14] F. Gemeinhardt, A. Garmendia, and M. Wimmer, "Towards model-driven quantum software engineering," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, 2021, pp. 13–15. [Online]. Available: https://doi.org/10.1109/Q-SE52541.2021.00010

[15] J. Campos and A. Souto, "Qbugs: A collection of reproducible bugs in quantum algorithms and a supporting infrastructure to enable controlled quantum software testing and debugging experiments," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, 2021, pp. 28–32. [Online]. Available: https://doi.org/10.1109/Q-SE52541.2021.00013

[16] P. Zhao, J. Zhao, and L. Ma, "Identifying bug patterns in quantum programs," in *2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)*, 2021, pp. 16–21. [Online]. Available: https://doi.org/10.1109/Q-SE52541.2021.00011

[17] J. Zhao, "Quantum software engineering: Landscapes and horizons," *CoRR*, vol. abs/2007.07047, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2007.07047

[18] M. Piattini, G. Peterssen *et al.*, "The talavera manifesto for quantum software engineering and programming." in *QANSWER*, 2020, pp. 1–5.

[19] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the nisq era," *Quantum Science and Technology*, vol. 5, no. 4, 2020. [Online]. Available: https://dx.doi.org/10.1088/2058-9565/abae7d

[20] C. Kai-Uwe Becker, N. Tcholtchev *et al.*, "Towards a quantum benchmark suite with standardized kpis," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022, pp. 160–163. [Online]. Available: https://doi.org/10.1109/ICSA-C54293.2022.00038

[21] P. G. Teague Tomesh, "Supermarq: A scalable quantum benchmark suite," *28th IEEE International Symposium on High-Performance Computer Architecture*, 2022. [Online]. Available: https://par.nsf.gov/biblio/10339323

[22] S. Resch and U. R. Karpuzcu, "Benchmarking quantum computers and the impact of quantum noise," *ACM Comput. Surv.*, vol. 54, no. 7, jul 2021. [Online]. Available: https://doi.org/10.1145/3464420

[23] P. J. Salas, "Noise effect on grover algorithm," *The European Physical Journal D*, vol. 46, no. 2, pp. 365–373, 2008. [Online]. Available: https://doi.org/10.1140/epjd/e2007-00295-1

[24] M. Alam, A. Ash-Saki, and S. Ghosh, "Design-space exploration of quantum approximate optimization algorithm under noise," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*, 2020, pp. 1–4. [Online]. Available: https://doi.org/10.1109/CICC48029.2020.9075903

[25] S. Wang, E. Fontana *et al.*, "Noise-induced barren plateaus in variational quantum algorithms," *Nature Communications*, vol. 12, no. 1, p. 6961, 2021. [Online]. Available: https://doi.org/10.1038/s41467-021-27045-6

[26] T. Hubregtsen, D. Wierichs *et al.*, "Training quantum embedding kernels on near-term quantum computers," *Phys. Rev. A*, vol. 106, Oct 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.106.042431

[27] M. Franz, L. Wolf *et al.*, "Uncovering instabilities in variational-quantum deep q-networks," *Journal of The Franklin Institute*, 8 2022. [Online]. Available: https://doi.org/10.1016/j.jfranklin.2022.08.021

[28] J. Liu, F. Wilde *et al.*, "Noise can be helpful for variational quantum algorithms," 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.06723

[29] J. D. Guimarães, J. Lim *et al.* (2023) Noise-assisted digital quantum simulation of open systems. [Online]. Available: https://doi.org/10.48550/arXiv.2302.14592

[30] C. W. Gardiner and P. Zoller, *Quantum noise: a handbook of Markovian and non-Markovian quantum stochastic methods with applications to quantum optics*, 2nd ed., H. Haken, Ed. Springer, 2000.

[31] K. Bharti, A. Cervera-Lierta *et al.*, "Noisy intermediate-scale quantum algorithms," *Rev. Mod. Phys.*, vol. 94, Feb 2022. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.94.015004

[32] S. Boixo, S. V. Isakov *et al.*, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, apr 2018. [Online]. Available: https://doi.org/10.1038/s41567-018-0124-x

[33] A. W. Cross, L. S. Bishop *et al.*, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, Sep 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.100.032328

[34] T. Lubinski, S. Johri *et al.*, "Application-oriented performance benchmarks for quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–32, 2023. [Online]. Available: https://doi.org/10.1109/TQE.2023.3253761

[35] T. Lubinski, C. Coffrin *et al.*, "Optimization applications as quantum performance benchmarks," *arXiv*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.02278

[36] T. Krüger and W. Mauerer, "Quantum annealing-based software components: An experimental case study with sat solving," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 445–450. [Online]. Available: https://doi.org/10.1145/3387940.3391472

[37] A. J. McCaskey, Z. P. Parks *et al.*, "Quantum chemistry as a benchmark for near-term quantum computers," *NPJ Quantum Information*, vol. 5, no. 1, p. 99, 2019. [Online]. Available: https://doi.org/10.1038/s41534-019-0209-0

[38] M. Schönberger, S. Scherzinger, and W. Mauerer, "Ready to leap (by co-design)? join order optimisation on quantum hardware," in *Proceedings of ACM SIGMOD/PODS International Conference on Management of Data*, 2023.

[39] A. Li, S. Stein *et al.*, "Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation," *ACM Transactions on Quantum Computing*, vol. 4, no. 2, feb 2023. [Online]. Available: https://doi.org/10.1145/3550488

[40] M. M. Koen Mesman, Zaid Al-Ars, "Qpack: Quantum approximate optimization algorithms as universal benchmark for quantum computers," *arXiv*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2103.17193

[41] J. R. Finžgar, P. Ross *et al.*, "Quark: A framework for quantum computing application benchmarking," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 226–237. [Online]. Available: https://doi.org/10.1109/QCE53715.2022.00042

[42] C. D. Bruzewicz, J. Chiaverini *et al.*, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, 2019. [Online]. Available: https://doi.org/10.1063/1.5088164

[43] I. Pogorelov, T. Feldker *et al.*, "Compact ion-trap quantum computing demonstrator," *PRX Quantum*, vol. 2, Jun 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PRXQuantum.2.020343

[44] "Ionq aria: Practical performance," https://ionq.com/posts/july-25-2022-ionq-aria-part-one-practical-performance, accessed: 2023-03-25.

[45] "Rigetti systems," https://qcs.rigetti.com/qpus, accessed: 2023-03-25.

[46] A. G. Fowler, M. Mariantoni *et al.*, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, sep 2012. [Online]. Available: https://doi.org/10.1103%2Fphysreva.86.032324

[47] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. [Online]. Available: https://doi.org/10.1017/CBO9780511976667

[48] K. Nakaji, S. Uno *et al.*, "Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicators," *Phys. Rev. Res.*, vol. 4, May 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.4.023136

[49] J. Liu and H. Zhou, "Hardware efficient quantum search algorithm," 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2103.14196

[50] M. Briański, J. Gwinner *et al.*, "Introducing structure to expedite quantum searching," *Phys. Rev. A*, vol. 103, Jun 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.062425

[51] H. Tezuka, K. Nakaji *et al.*, "Grover search revisited: Application to image pattern matching," *Phys. Rev. A*, vol. 105, Mar 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.105.032440

[52] A. Abbas, S. Andersson *et al.*, "Learn quantum computation using qiskit," 2020. [Online]. Available: https://qiskit.org/textbook/

[53] M. Zajac and U. Störl, "Towards quantum-based search for industrial data-driven services," in *2022 IEEE International Conference on Quantum Software (QSW)*, 2022, pp. 38–40. [Online]. Available: https://doi.org/10.1109/QSW55613.2022.00021

[54] H. Amellal, A. Meslouhi, and A. E. Allati, "Processing unstructured databases using a quantum approach," in *Innovations in Smart Cities Applications Edition 2*, M. Ben Ahmed, A. A. Boudhir, and A. Younes, Eds. Cham: Springer International Publishing, 2019, pp. 275–285. [Online]. Available: https://doi.org/10.1007/978-3-030-11196-0_25

[55] C. Figgatt, D. Maslov *et al.*, "Complete 3-qubit grover search on a programmable quantum computer," *Nature communications*, vol. 8, no. 1, pp. 1–9, 2017. [Online]. Available: https://doi.org/10.1038/s41467-017-01904-7

[56] K. Mitarai, M. Negoro *et al.*, "Quantum circuit learning," *Physical Review A*, vol. 98, no. 3, sep 2018. [Online]. Available: https://doi.org/10.1103%2Fphysreva.98.032309

[57] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, 2019. [Online]. Available: https://doi.org/10.1002/qute.201900070

[58] K. Kraus, A. Böhm *et al.*, *States, Effects, and Operations Fundamental Notions of Quantum Theory: Lectures in Mathematical Physics at the University of Texas at Austin*. Springer, 1983. [Online]. Available: https://doi.org/10.1007/3-540-12732-1

[59] M. A. Nielsen, "A simple formula for the average gate fidelity of a quantum dynamical operation," *Physics Letters A*, vol. 303, no. 4, pp. 249–252, 2002. [Online]. Available: https://doi.org/10.1016/S0375-9601(02)01272-0

[60] A. Gilchrist, N. K. Langford, and M. A. Nielsen, "Distance measures to compare real and ideal quantum processes," *Phys. Rev. A*, vol. 71, Jun 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.71.062310

[61] T. H. Artur Ekert, "Introduction to quantum information science," 2022. [Online]. Available: https://qubit.guide/qubit_guide.pdf

[62] K. Georgopoulos, C. Emary, and P. Zuliani, "Modeling and simulating the noisy behavior of near-term quantum computers," *Phys. Rev. A*, vol. 104, Dec 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.104.062432

[63] C. Blank, D. K. Park *et al.*, "Quantum classifier with tailored quantum kernel," *npj Quantum Information*, vol. 6, no. 1, p. 41, 2020, supplementary information. [Online]. Available: https://doi.org/10.1038/s41534-020-0272-6

[64] Qiskit-Development-Team, "Qiskit source-code: basic_device_gate_errors," 2023, version 0.12.0, Accessed: 2023-06-03. [Online]. Available: https://qiskit.org/ecosystem/aer/stubs/qiskit_aer.noise.device.basic_device_gate_errors.html

[65] E. Magesan, "Depolarizing behavior of quantum channels in higher dimensions," 2010. [Online]. Available: https://doi.org/10.48550/arXiv.1002.3455

[66] H. Pashayan, J. J. Wallman, and S. D. Bartlett, "Estimating outcome probabilities of quantum circuits using quasiprobabilities," *Phys. Rev. Lett.*, vol. 115, Aug 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.115.070501

[67] M. Cramer, M. B. Plenio *et al.*, "Efficient quantum state tomography," *Nature Communications*, vol. 1, no. 1, p. 149, 2010. [Online]. Available: https://doi.org/10.1038/ncomms1147

[68] S. Ebadi, T. T. Wang *et al.*, "Quantum phases of matter on a 256-atom programmable quantum simulator," *Nature*, vol. 595, no. 7866, pp. 227–232, 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03582-4

[69] K.-N. Schymik, V. Lienhard *et al.*, "Enhanced atom-by-atom assembly of arbitrary tweezer arrays," *Phys. Rev. A*, vol. 102, Dec 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.102.063107

[70] K. Wintersperger, H. Safi, and W. Mauerer, "Qpu-system co-design for quantum hpc accelerators," in *Architecture of Computing Systems*, M. Schulz, C. Trinitis *et al.*, Eds. Cham: Springer International Publishing, 2022, pp. 100–114. [Online]. Available: https://doi.org/10.1007/978-3-031-21867-5_7

[71] K. R. Brown, J. Kim, and C. Monroe, "Co-designing a scalable quantum computer with trapped atomic ions," *npj Quantum Information*, vol. 2, no. 1, p. 16034, Nov. 2016. [Online]. Available: https://doi.org/10.1038/npjqi.2016.34

[72] G. Li, A. Wu *et al.*, "On the co-design of quantum software and hardware," in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, ser. NANOCOM '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3477206.3477464

[73] S. Feld, C. Roch *et al.*, "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer," M. S. Sarandy, Ed., vol. 6. Frontiers Media SA, 6 2019. [Online]. Available: https://doi.org/10.3389/fict.2019.00013

[74] H. Safi, K. Wintersperger, and W. Mauerer, "Influence of hw-sw-co-design on quantum computing scalability," in *Proceedings of the IEEE Quantum Software Week*, 2023.

[75] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. [Online]. Available: https://doi.org/10.1017/CBO9780511813603

[76] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, Jul 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.031027

[77] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, 2021. [Online]. Available: https://doi.org/10.1038/s41567-021-01287-z