A Performance Study of Variational Quantum Algorithms for Solving the Poisson Equation on a Quantum Computer

Mazen Ali\textsuperscript{1,\ast} and Matthias Kabel\textsuperscript{1,†}

\textsuperscript{1}Fraunhofer ITWM, 67663 Kaiserslautern, Germany

Recent advances in quantum computing and their increased availability has led to a growing interest in possible applications. Among those is the solution of partial differential equations (PDEs) for, e.g., material or flow simulation. Currently, the most promising route to useful deployment of quantum processors in the short to near term are so-called hybrid variational quantum algorithms (VQAs). Thus, variational methods for PDEs have been proposed as a candidate for quantum advantage in the noisy intermediate scale quantum (NISQ) era. In this work, we conduct an extensive study of utilizing VQAs on real quantum devices to solve the simplest prototype of a PDE – the Poisson equation. Although results on noiseless simulators for small problem sizes may seem deceivingly promising, the performance on quantum computers is very poor. We argue that direct resolution of PDEs via an amplitude encoding of the solution is not a good use case within reach of today’s quantum devices – especially when considering large system sizes and more complicated non-linear PDEs that are required in order to be competitive with classical high-end solvers.

I. INTRODUCTION

The technological progress in quantum computing has spurred a lot of research into applications with the potential for an advantage over classical computing. One of these possible applications is the solution of partial differential equations (PDEs) that are extensively used in various areas of engineering such as computational fluid dynamics (CFD)\textsuperscript{11} or material simulation\textsuperscript{2,3}.

In its simplest form, a linear PDE is transformed via a discretization method into a system of linear equations. The latter can then be solved with the quantum HHL method\textsuperscript{[1]}. For general linear PDEs in three spatial dimensions, one can expect at best a quadratic speedup compared to classical solvers, see\textsuperscript{[4]}. The speedup may, however, increase for high-dimensional PDEs. Despite recent progress with HHL\textsuperscript{[6–8]}, it requires deep entangling circuits and has thus limited scalability within the noisy intermediate scale quantum (NISQ) era computers.

A more NISQ friendly alternative are so-called (hybrid) variational algorithms. These are (mostly) heuristic methods that rely on a quantum-classical approach where the quantum computer is only used to execute relatively shallow circuits to estimate cost functionals within a classical optimization loop. Hybrid methods have gained a lot of attention\textsuperscript{[9]} as the class of methods for NISQ devices, including applications to PDEs\textsuperscript{[10,12]}.

The simplest example of a model PDE problem is the 2nd order linear Poisson equation. This has been previously addressed in\textsuperscript{[10,11]}

\textsuperscript{[13]}using the hybrid variational quantum linear solver (VQLS) from\textsuperscript{[13]} with tests on simulators. In this work, we conduct a thorough study of the applicability of variational hybrid methods to PDEs by performing extensive tests with VQLS for the Poisson equation on both simulators with statistical finite sampling (shot) noise and real quantum hardware using superconducting qubits\textsuperscript{[13]}.

Our results indicate that hybrid solvers for PDEs are not a promising route for achieving quantum advantage in the short to near term. It is well known that, when increasing system size, a) PDEs require preconditioning and b) variational algorithms suffer from barren plateaus. Neither of these issues are present for the system sizes considered in this work. Nonetheless, VQLS struggles to converge (fast) even for small system sizes – both on quantum hardware and simulators with shot noise – in the absence of a), b), data encoding and readout issues. Moreover, the competition – namely classical PDE solvers – can typically achieve precision that seems beyond the reach of quantum algorithms relying on finite sampling.

A common argument for the potential of quantum advantage is the scalability with respect to the number of spatial dimensions of the PDE – ignoring data encoding and decoding, there is no apparent “curse of dimensionality” in VQAs applied to, e.g., the Poisson equation\textsuperscript{[13]}. However, for interesting system sizes of \(n \geq 20\) qubits and the resulting increase in noise, we believe other classical methods for high-dimensional PDEs\textsuperscript{[16–18]} are more promising than VQAs. While there still may be some benefit in using NISQ devices for, e.g., material simulation or CFD, we believe it does not lie within the direct resolution of PDEs via an amplitude encoding of the solution (see Section\textsuperscript{[11]}).

Finally, we mention that in this work we considered several but not all error mitigation techniques. Particularly noteworthy is probabilistic error cancellation (PEC)\textsuperscript{[19,20]} that was recently added to IBM’s runtime service. PEC attempts to produce unbiased estimates of expectation values by fitting a (sparse) Pauli noise model to the physical noise on the quantum device and implementing the inverse of said noise channels by sampling randomized Pauli twirled circuits. The success of this denoising method depends on several non-trivial assumptions.

\textsuperscript{[13]}mazen.ali@itwm.fraunhofer.de
\textsuperscript{[14]}matthias.kabel@itwm.fraunhofer.de
about the quality of the noise model and involves an exponential sampling overhead. It is a potentially interesting research question whether PEC would benefit the estimation of observables described in this work and if it would scale to large system sizes.

The remainder of the paper is organized as follows. In Section II we introduce the Poisson equation and the different ways to estimate the Poisson operator on a gate based quantum computer. In Section III we go over some basics of VQLS and the different cost functions used for optimization. In Section IV we test individual components of VQLS for the Poisson equation – both on simulators and quantum hardware – and conclude with the overall performance of the VQLS optimization in the presence of noise.

II. POISSON EQUATION

The Poisson equation with Dirichlet boundary conditions is defined as

\[-\nabla^2 u(x) = f(x) \quad \text{for } x \in \Omega,\]

\[u(x) = 0 \quad \text{for } x \in \partial \Omega,\]

for an open domain \( \Omega \subset \mathbb{R}^d \) with Lipschitz boundary \( \partial \Omega \) and real-valued \( u : \Omega \to \mathbb{R} \) and \( f : \Omega \to \mathbb{R} \). The weak (variational) form of this equation reads: find \( u \in H^1_0(\Omega) \) that satisfies

\[a(u, v) := \int_{\Omega} \nabla u(x) \cdot \nabla v(x) \, dx = \int_{\Omega} f(x)v(x) \, dx =: f(v),\]

for all \( v \in H^1_0(\Omega) \), where \( H^1_0(\Omega) \) is the space of all weakly differentiable \( L^2 \)-functions that are zero on \( \partial \Omega \) in the trace sense.

In the finite element method (FEM), one discretizes the Poisson equation by choosing a set of basis (test and trial) functions

\[V_h := \{ \varphi_k : k = 0, \ldots, N - 1 \} \subset H^1_0(\Omega),\]

and, consequently, the discrete Poisson equation reads: find \( u = (u_k)_{k=0}^{N-1} \) such that \( u_k := \sum_{j=0}^{N-1} u_{kj} \varphi_j \) and

\[a(u_k, v_k) = f(v_k), \quad \text{for all } v_k \in V_h,\]

or, equivalently, solve

\[Au = f,\]

\[A := (a(\varphi_j, \varphi_i))_{i,j=0}^{N-1},\]

\[f := (f(\varphi_i))_{i=0}^{N-1}.\]

In this work, we test the simplest case \( d = 1, \Omega = (0, 1) \) and piecewise linear FEM. Therefore, \( A \) is a tridiagonal matrix with 2’s on the main diagonal and -1’s on the off-diagonals

\[A = \frac{1}{h} \begin{pmatrix}
2 & -1 & 0 & \cdots \\
-1 & 2 & -1 & \cdots \\
& & \ddots & & \ddots \\
& & & -1 & 2
\end{pmatrix},\]

where \( h = \frac{1}{N+1} \) denotes the mesh size (distance between discretization nodes). Alternatively, one can re-scale the domain \( \Omega \) to \((0, N + 1)\), in which case \( h = 1 \).

For the right-hand-side (RHS) \( f \), we consider the two border cases where \( f \) is constant and where \( f \) contains a discontinuous jump. The constant case leads to a smooth solution \( u \) whereas the discontinuous case leads to a singularity (in the derivatives) of \( u \). While this significantly impacts the performance of classical methods (or requires special techniques such as, e.g., adaptive or hp-methods), the discontinuity has no substantial effect on the training in Section IV. Both cases can be approximated as quantum states on \( n \) qubits with the following unitaries

\[|f_C\rangle := H^\otimes n |0\rangle_n, \quad (H_n)\]

\[|f_D\rangle := H^\otimes n - 1 \otimes X |0\rangle_n, \quad (H_nX)\]

where \( H \) and \( X \) are the Hadamard and Pauli-\( X \) gates, respectively.

We cast the LSE problem from \([1]\) to a problem of determining a quantum state via an amplitude encoding as follows. Find an \( n \)-qubit quantum state \(|u\rangle = \frac{1}{\|u\|} \sum_{k=0}^{N-1} u_k |k\rangle\) that satisfies

\[\langle u|A|u\rangle = \langle f|f\rangle.\]

We re-scale such that \( h\|f\| = 1 \) for simplicity and since this does not affect conclusions about the overall performance of VQLS.

Within a variational solver, one uses a parametrized ansatz \(|\psi(\theta)\rangle\) and optimizes over the parameters \( \theta = (\theta_1, \ldots, \theta_p) \) to approximate the normalized solution \(|u\rangle \approx |\psi(\theta^*)\rangle\). To this end, we must estimate terms such as \( \langle \psi(\theta)|A|\psi(\theta)\rangle \) on a quantum computer. There are several ways of estimating the observable \( A \): details of all decompositions used in this work are provided in Fig. 14 – 17.

The simplest option is decomposing \( A \) into Pauli strings. This requires \( O(2^n) \) terms and, consequently, any potential quantum advantage is lost. Another option was presented in \([11]\), where \( A \) is decomposed into \( O(n) \) simple operators, see Fig. 16. Note that, however, the decomposition terms in Liu21 commute and can be thus grouped together into \( O(1) \) terms, see Fig. 17. Finally, in \([10]\), yet another decomposition method was proposed with \( O(1) \) terms.

While all of these decompositions are mathematically equivalent, they have a trade-off between number of circuits to run vs. entangling gates required for each circuit.
The matrix $A$ from (2) can be decomposed as

$$A = 2I^\otimes n - I^\otimes n-1 \otimes X + R. \quad (4)$$

The expectation of the first term is constantly equal to 2. The second term requires adding only one Hadamard gate on the least significant qubit [21], i.e., here the accuracy will mostly depend on the gates required for the ansatz $|\psi(\theta)\rangle$. The remaining term $R$ (off-diagonal $-1$’s on odd positions) requires either exponentially many Pauli strings or one highly entangling circuit, see also Fig. 15. On one hand, using highly entangling unitaries, the estimation of the expectation value of $R$ is inaccurate on today’s quantum computers. On the other hand, as we will see in Section IVD, the expectation value of $A$ is dominated by the constant part. This is a good example of an abstract mathematical problem that is deceivingly simple to solve classically, but is rather complicated to run on a modern day quantum computer.

### III. VQLS COST FUNCTIONS

VQLS was first introduced in [13]. It is an extension of the variational quantum eigensolver [22], where one uses a parametrized ansatz function $|\psi(\theta)\rangle$ and minimizes the expected energy of some Hamiltonian. The different types of ansatz used in this work are detailed in Fig. 12.

For linear systems, the authors in [13] propose Hamiltonians for which the ground state corresponds to the normalized solution of (3), the estimation accuracy and convergence of optimizers varies. Although the global minimum of all four cost functions corresponds to the normalized solution of (3), the expectation value of $\langle \psi(\theta) | H | \psi(\theta) \rangle$ is its normalized version, i.e., $s^2$ stands for the norm of $u$. Moreover, we note that the cost functions $C_N$ and $C_{NN}$ require fewer and simpler circuits than $C_G$ and $C_L$.

### IV. RESULTS

A VQLS setup consists of choosing the type of ansatz, the RHS, the number of qubits and layers of the ansatz, the cost function, the classical optimization method and the estimation backend. In Section IV A we probe the expressivity and trainability of different ansatze and cost functions by testing on noiseless state-vector simulators, i.e., in the absence of both hardware noise and statistical finite sampling (shot) noise. In Section IV B we test the fidelity of different ansatze both on simulators with shot noise and on different IBM backends. In Section IV C we test the accuracy of estimating inner products. In Section IV D we test the accuracy of estimating the Poisson operator expectation. In Section IV E we test the accuracy of estimating different cost functions. In Section IV F we test the cosine similarity of estimating gradients. Finally, in Section IV G we test the overall performance of several optimizers with different cost functions. The tests in Sections IV B – IV F were performed on a sample of randomly generated ansatz parameters, where we compared, e.g., different error percentiles.

The solution fidelity is defined as

$$F(\theta) := | \langle \psi(\theta) | u \rangle |^2,$$

where $|u\rangle$ is the exact normalized solution. We always compute the solution fidelity numerically exact, i.e., whether $\theta$ was optimized on simulators or a real quantum device, the above fidelity is computed using the exact vector for $|u\rangle$ and the exact vector for $|\psi(\theta)\rangle$.

### A. Training Without Noise

In this section, we estimate all quantities on a state-vector simulator, i.e., numerically exact and choose the BFGS optimizer [25,26]. It was observed both in previous work [23] and our own tests that BFGS performs best on small variational problems in the absence of noise. We run each VQLS setup for 15 randomly generated initial values and select the best run based on the smallest observed cost function value. We use the same set of random initial values across all tests.
Figure 1: Solution fidelity vs. number of cost function evaluations. Note that we display all cost evaluations, i.e., not only cost for current (accepted) iterate \( \theta_k \). RHS (HnX), BFGS optimizer, state vector simulator (no noise). (a) Ansatz linear alternating \( R_y-CZ \) (see Fig. 12), \( n = 3, l = 0 \). (b) Ansatz linear alternating \( R_y-CZ \), \( n = 3, l = 3 \). (c) Ansatz linear alternating \( R_y-CZ \), \( n = 5, l = 3 \). (d) Ansatz linear alternating \( R_y-CZ \), \( n = 8, l = 3 \). (e) Ansatz linear alternating \( R_y-CZ \) ansatz, RHS (Hn), \( n = 5, l = 3 \). (f) Final solution fidelity vs. number of ansatz parameters, RHS (HnX), \( n = 5 \), \( R_y-CZ \) ansatz, linear vs linear alternating.

In Fig. 1 we display the solution fidelity for the minimization based on the four different cost functions from 5, for a varying number of qubits and ansatz layers. The cost function \( C_N \) leads to the fastest convergence. The non-normalized version \( C_{NN} \) displays similar but slightly worse behavior. Our main reason for considering \( C_{NN} \) is a better behavior in the presence of noise, as we will see in the following sections. The global and local cost func-
tions, \( C_G \) and \( C_L \), fail to converge at a relatively small number of qubits and are, in addition, more costly to estimate than \( C_N \) and \( C_{NN} \).

The choice of the RHS does not pose a problem for the \( C_N \) and \( C_{NN} \) cost functions. For \( C_G \) and \( C_L \), the barren plateaus worsen for the RHS (\( H_{\theta} \)). This may seem counter-intuitive from the perspective of classical PDE methods, where smoother data is generally associated with better convergence. For VQLS, however, particular combinations of ansatze and RHSs can lead to very flat optimization landscapes or even constant cost functions. The choice of alternating vs. non-alternating entanglement pattern from Fig. 12 has a significant impact on the final solution fidelity.

### B. Ansatz Fidelity

Having computed the optimal parameters, how accurately can we sample the corresponding distribution from the quantum device? In this section, we test the fidelity of different ansatze across multiple backends. As a benchmark, we will use the sampling error of a simulator with shot noise. For the error metric, we use the Hellinger fidelity of two distributions \( p \) and \( q \)

\[
F(p, q) := \left( \sum_i \sqrt{p_i q_i} \right)^2 = |\langle \psi(\theta) | \psi(\theta) \rangle|^2,
\]

where \( p_i = |\langle i | \psi(\theta) \rangle|^2 \) and \( q_i = |\langle i | \psi(\theta) \rangle|^2 \) is the true ansatz state and \( |\psi(\theta)\rangle \) is the ansatz state prepared on a real quantum device. More precisely, we have access only to an estimate \( \hat{q}_i \) of the true probability \( q_i \). The different ansatz architectures are described in Fig. 12.

In Fig. 2a, we see that the median fidelity is quite close to the noiseless quantum simulator up to 10 qubits. Adding entangling layers at 10 qubits reduces the fidelity significantly. Comparing the different backends, the performance is similar, with dips in fidelity observed on either newer backends or on small backends where we exhausted the maximum number of qubits.

In Fig. 2b, we compare different ansatz architectures. Some results are to be expected, e.g., adding periodic entanglement (CNOTs on first and last qubit) leads to more CNOTs post transpilation – since the topology of IBM devices does not natively accommodate such entanglement (see Fig. 18) – and, hence, lowers fidelity. Other results are, however, less intuitive. For example, linear \( R_X \)-CX performs slightly better than linear \( R_Z \)-CX, even though \( R_Z \) rotations are native to IBM hardware. More notably, linear \( U3-CX \) (all 3 rotations) performs best despite performing more rotations. The number of parameters is tripled for \( U3 \), i.e., for \( p = 180 \), \( U3-CX \) has as many entangling layers as \( R_Y-CZ \) for \( p = 60 \).

In Fig. 2c, we consider the fidelity behavior when increasing the number of shots (measurement samples). The convergence on a noiseless quantum simulator and ibmq_ehningen are similar with a constant offset due to hardware noise. For the simulator, the number of entangling layers does not affect the fidelity, as expected, whereas for the real backend adding layers accumulates hardware error.

Finally, when executing circuits on a real backend, one has various transpilation and error mitigation options. For comparison, we select those options that we consider to be relatively generic and easy to run by most users. For transpilation, we compare no optimization with Qiskit’s swap-optimized transpilation based on the SABRE heuristic. Due to the stochastic nature of this heuristic, we transpile 20 times and select the circuit minimizing a weighted average of depth and number of CNOTs, with number of CNOTs having twice the weight. For error mitigation (EM), we test either no EM, matrix-free measurement EM (MEM) \( \text{MEM} \), simple \( XX \) dynamic decoupling (DD) with as-late-as-possible instruction scheduling or noise aware transpilation (the mapomatic tool from \( \text{MEM} \)). The results are summarized in Fig. 2d.

Optimized transpilation and measurement error mitigation consistently improve fidelity. When an algorithm does not change the topology of input circuits, transpilation has to be performed only once and, thus, the overhead is negligible. For MEM, one has to perform regular calibrations on the target backend. If integrated into the output pipeline, the overhead for MEM can be negligible as well.

The results for dynamic decoupling are less clear. From Fig. 2d and other tests not illustrated here, we observed that it can both improve and worsen the results. We are not aware of a universal DD method that works on any circuit, it is still an active area of research and a more problem-tailored DD method might be required for consistent improvement (see, e.g., \( \text{MEM} \)).

The same conclusion applies to noise aware transpilation \( \text{MEM} \). Here, a better choice of cost function than the default for selecting less noisy qubits might improve results. We do not investigate this further.

### C. Inner Products

Estimating inner products is a basic numerical subroutine required by any linear solver. In our case, we estimate inner products of the form \( \Re\langle f | \psi(\theta) \rangle \). This is typically done via the Hadamard test as in \( \text{10, 13} \). For the special case of the cost function \( C_N \) from \( \text{(5)} \) and an ansatz with only real-valued amplitudes, we can estimate \( |\Re\langle f | \psi(\theta) \rangle|^2 \) by measuring the overlap, which requires fewer controlled gates than the Hadamard test. The overlap test measures all qubits as opposed to one qubit for the Hadamard test.

We compare relative errors since the exact value of \( \Re\langle f | \psi(\theta) \rangle \) is often quite small and thus small absolute errors are not informative. Let \( \mathcal{E}(\theta) := \mathcal{E}(\Re\langle f | \psi(\theta) \rangle) \) denote the estimate of the inner product computed by sampling on a simulator or a real quantum device. The
The exact definition of the relative error depends on whether we compare the Hadamard and overlap test in the same plot, or only the Hadamard test. For the former, the relative error is defined as

$$\epsilon_{\text{rel}}^2(\theta) := \left| \frac{\langle f | \psi(\theta) \rangle^2 - \mathcal{E}(\theta)^2}{|\langle f | \psi(\theta) \rangle|^2} \right|,$$

and for the latter

$$\epsilon_{\text{rel}}(\theta) := \frac{|\mathcal{R} \langle f | \psi(\theta) \rangle | - \mathcal{E}(\theta)|}{|\mathcal{R} \langle f | \psi(\theta) \rangle |}.$$ 

This is simply because, when using the overlap test, we only have access to the squared quantity and thus a direct comparison of $\mathcal{E}(\theta)$ with $\mathcal{E}(\theta)^2$ in the same plot is not meaningful.

In Fig. 3a, we compare the different methods w.r.t. the number of layers and shots with references to the exact value $|\langle f | \psi(\theta) \rangle|^2$. The conclusions are different depending on whether one considers only the median or the mean. The overlap test performs slightly better if one ignores outliers, and vice versa for the Hadamard test. Therefore, on today’s quantum computing hardware, the additional overhead for adding QAE to each subroutine call is substantial for the overall training, while the accuracy gain is potentially negligible.

Finally, in Fig. 3c we consider adding different error mitigation techniques for the Hadamard test. Unlike in Section IVB, the mean relative error is smallest when applying all error mitigation techniques, while apply-
Figure 3: Relative error inner product estimation. 50 parameter samples in all plots. (a) Median relative error vs. number of layers, \( n = 5 \) qubits, linear \( R_Y - C_Z \) ansatz, RHS \( \{H_n\} \), 1000 shots. (b) Median relative error vs. number of shots, 5 qubits and 3 layers, linear \( R_Y - C_Z \) ansatz, RHS \( \{H_n\} \), 1000 shots. (c) Same as (a) but mean instead of median. (d) Same as (b) but mean instead of median. (e) Mean relative error vs. number of layers for the Hadamard test for different QAE methods, \( R_Y - C_Z \) ansatz, 5 qubits, RHS \( \{H_n\} \). (f) Mean relative error vs. number of layers for the Hadamard test for different error mitigation techniques, \( R_Y - C_Z \) ansatz, 5 qubits, RHS \( \{H_n\} \).

Although there is a lot of variability in errors depending on ansatz architecture, RHS, number of layers and error mitigation techniques – in all cases the relative error for estimating the basic quantity \( \Re \langle f | \psi(\theta) \rangle \) is very large, even on simulators. Furthermore, inner product estimation is a basic subroutine that would be required to...
estimate functionals of the solution such as, e.g., taking the average of $u$, and this is a necessary step to gain any sort of quantum advantage for PDE solvers [5]. Based on our tests, even if the solution $u$ is accurately approximated on a quantum device via $|\psi(\theta)\rangle$, the necessary accuracy for estimating $\langle r|\psi(\theta)\rangle$ for some functional $r$ and $n \geq 20$ qubits translate into a very large number of shots together with high requirements for hardware fidelity and error mitigation that may be unrealistic in the near term.

D. Operator Expectation

In this section, we test the accuracy of estimating $\langle \psi(\theta)|A|\psi(\theta)\rangle$. As discussed in Section [11], there are at least four different ways to estimate $A$, refer to Fig. [14 – 17] for the different decompositions.

In Fig. [4], we compare the relative errors defined as

$$\epsilon_{rel}(\theta) := \frac{\langle \psi(\theta)|A|\psi(\theta)\rangle - E(\theta)}{\langle \psi(\theta)|A|\psi(\theta)\rangle},$$

where we now abbreviate the estimate computed by sampling on a simulator or a real quantum device as $E(\theta) := \langle \psi(\theta)|A|\psi(\theta)\rangle$. At first glance, this estimation seems to perform much better than inner product estimation from Section [IVC]. However, this is mainly due to the “constant” part of $A$, see (4), $\langle \psi(\theta)|A|\psi(\theta)\rangle \approx 2$.

Fig. [4] shows that most of the error is hardware noise, e.g., for $q = 5$, $l = 3$ it is not worth going beyond 100 – 1000 shots. Fig. [4b] shows the number of CNOTs required for the different decomposition methods. Together with Fig. [4c] the Liu21 decomposition (see Fig. [16]) performs best. We do not consider using the Pauli decomposition a viable option, since it does not scale. Recall that the number of observables for Liu21 scales linearly with the number of qubits as opposed to constant for Sato21 and Liu21Grouped.

E. Cost Functions

In this section, we compare the accuracy of estimating $C_N$ and $C_{NN}$ from [5]. For inner product estimation, we use the Hadamard test, and, for operator expectation, we use the Liu21 decomposition. For a cost function $C(\theta)$ and its estimate $\hat{C}(\theta)$, the absolute and relative errors are defined as

$$\epsilon_{abs}(\theta) := |C(\theta) - \hat{C}(\theta)|,$$

$$\epsilon_{rel}(\theta) := \epsilon_{abs}(\theta)/|C(\theta)|.$$

In Fig. [5a], we compare the accuracy w.r.t. the number of shots. Unlike before, we show a more detailed comparison here, including the $p = 0, 5, 25, 50, 75, 95, 100$ error percentiles. We see that the accuracy saturates at about $S = 1000$ shots. Moreover, estimation of $C_{NN}$ is slightly more accurate.

Next, we test if the mean absolute error in cost function estimation is larger than the mean variation of exact cost function values. This should indicate if an optimizer can detect a descent direction in the presence of hardware noise. We estimate the variation as follows. Generate a sample of 50 random parameters $\Omega := \{\theta^1, \ldots, \theta^{50}\}$. Then, for each parameter in $\Omega$, sample 100 random parameters $\Delta(\theta) := \{\delta^1(\theta), \ldots, \delta^{100}(\theta)\}$, where each of the components of $\delta^k(\theta)$ is between $-2\pi$ and $2\pi$. Select a step size and compute the variations

$$\text{var}(\theta, \delta) := |C(\theta) - C(\theta + \text{step} \cdot \delta)|,$$

for each $\delta$ in $\Delta(\theta)$ and each $\theta$ in $\Omega$. In total, we thus have differences of cost function values sampled at 500 different points.

The results are summarized in Fig. [5b – 5d]. In both cases the estimation is quite noisy as the absolute error in cost function value is within the range of cost function variation. Thus, in general, it is unclear if an optimizer can recognize a descent direction in such a noisy regime.

The results for $C_{NN}$ are slightly better. This is mostly due to the norm parameter $s$ in [6]. We thus expect that, for random initial values, an optimizer would at first perform better for $C_{NN}$. However, as the optimization progresses and the value of $s$ is improved, $C_{NN}$ should encounter the same noise issues as $C_N$.

F. Gradients

In this section, we test the cosine similarity of estimated vs. exact gradients for $C_N$ and $C_{NN}$. The cosine similarity of two vectors $x$ and $y$ is defined as $\langle x|y\rangle/\langle x||y\rangle$. The results are summarized in Fig. [6].

Per optimizer iteration, often several calls of gradient estimation are required. For each gradient call, the number of circuits to run is $(C_1 + C_2)p$, where $S$ is the number of shots, $p$ is the number of ansatz parameters, $C_1$, and $C_2$ are constants depending on the cost function estimation method. For, e.g., the Liu21 decomposition, $C_1 = n + 1$ and $C_2 = 2n + 1$. On today’s quantum computers, this is a considerable overhead. Thus, for the results in Fig. [6b] on a real backend, we only use one ansatz layer and 10 parameter samples (in total 1490 circuits to run $S$ times each). As we can see, the gradient accuracy on ibmq_montreal is very poor for $C_N$ and much better for $C_{NN}$. For the latter, we will see in the next section whether this is sufficient for an optimizer to find a good solution.

G. Training With Noise

In this section, our VQLS setup consists of a linear alternating $R_Y$-$CZ$ ansatz with $n = 5$ qubits and $l = 3$ layers ($p = 29$ parameters), RHS $\text{H}_{\text{HNS}}$. We have seen in Section [IVC] that in this setting the ansatz is capable
of representing the solution with near to 100% fidelity. We use the Hadamard test to estimate inner products for both $C_N$ and $C_{NN}$, and the Liu21 decomposition to estimate $\langle \psi(\theta) | A | \psi(\theta) \rangle$. We run the tests on a simulator with shot noise and the ibmq_ehningen backend. We transpile all circuits as mentioned in Section IVB. For results on the quantum device, we present different combinations of transpilation and measurement error mitigation options. Note that we display the fidelity for all parameters $\theta$ at which the cost was evaluated, i.e., not only for accepted iterates $\theta_k$.

There is a plethora of classical optimizers one could use, many are implemented in Qiskit (most are wrappers for the SciPy package). Based on the results of Section IV A other works [29, 33, 37] and our own tests, we only present here the BFGS [25–28], SPSA [38], Powell [39] and NFT [35] optimizers. Other optimizers do not seem to perform better in the presence of hardware noise. Another possible noise-robust candidate – that we did not test here – is the Bayesian optimizer from [40].

In Fig. 7–10 we summarize our findings. BFGS performs significantly better for $C_{NN}$, achieving over 98% fidelity for 10000 shots on a simulator. On a quantum computer, for $C_{NN}$, BFGS achieves roughly 17% fidelity but does not improve after that. Moreover, the additional cost of evaluating gradients (see also Section IVF) is very time consuming on current quantum hardware: for the linear alternating $R_Y$-$CZ$ ansatz on 5 qubits with 3 layers and the Liu21 decomposition for $A$, one has to run 325 circuits $S$ times to estimate one gradient. The few iterations as displayed in Fig. 7 require hours to execute on ibmq_ehningen. Other gradient based methods implemented in Qiskit failed to converge even with shot noise only (for a moderate number of shots $S \leq 10^5$).

SPSA does not converge with the default choice of hyperparameters, i.e., using the calibration procedure from [37]. Setting the learning rate to 1, perturbation to 0.1, blocking and trust region to true, we obtain the convergence displayed in Fig. 8. We were not able to find a hyperparameter choice that works on a quantum computer. With QN-SPSA [41], we obtained similar results. SPSA convergence might be improved by an adaptive
hyperparameter selection as in, e.g., [12], but we do not investigate this further.

Finally, in Fig. [9] and [10] we present the results for Powell and NFT, respectively. The only tunable hyperparameter for NFT is the reset interval: a smaller interval generally leads to smoother convergence but more cost function evaluations. In this example, we set the reset interval of NFT to 9. On simulators, NFT trained with \( C_{\text{NN}} \) performed best, achieving almost 98% fidelity with only 1000 shots per circuit. A similar fidelity was obtained with BFGS trained on \( C_{\text{NN}} \) with 10000 shots per circuit, at the expense of additionally estimating gradients. On ibmq_ehningen, NFT trained on \( C_{\text{NN}} \) also performed best. Although the maximum observed fidelity was over 80%, the fidelity corresponding to the best observed cost value is only 10.89% (\( C_{\text{N}} \) trained with no EM). Similar observations about the noise robustness of NFT were made in [35, 36], however, overall, all trainings on quantum devices performed very poorly.

As discussed in the Introduction and Section IV C we are ultimately interested in estimating inner products with linear functionals. To that end, a more appropriate error metric would be the trace distance. Since we are dealing with pure quantum states, there is a simple relationship between the two

\[
\frac{1}{2} \text{tr} \left( \left| \langle u | - \langle \psi(\theta) | \psi(\theta) \rangle \right| \right) = \sqrt{1 - F(|u\rangle, |\psi(\theta)\rangle)},
\]

where \( F(|u\rangle, |\psi(\theta)\rangle) \) is the fidelity between the two pure states. Consequently, a fidelity of 10.89% translates into a trace distance of 0.944, which is a very large error. For convenience, we plot the relationship between fidelity and trace distance in Fig. [11]. For a reasonable trace error of \( \leq 10^{-1} \), one would require a solution fidelity of at least 99%. This does not include the error of estimating the inner product \( \langle r | \psi(\theta) \rangle \) itself.

**V. CONCLUSION**

We conducted a thorough investigation of the feasibility of applying modern-day quantum computers to partial differential equations. Specifically, we have extensively tested the variational quantum linear solver for
the simple Poisson problem using IBM’s superconducting quantum devices.

Firstly, we stress that even a noiseless quantum computer introduces significant errors into basic subroutines such as the inner product computation due to finite sampling. Nonetheless, as we demonstrated in this simple example, a good optimizer may still find a good solution for a moderate number of shots.

Secondly, estimating expectation values of differential operators involves a trade-off between the number of circuits to run and the number of 2-qubit entangling gates in each circuit. A scalable variant of the algorithm necessarily involves a large number of 2-qubit gates.

Thirdly, the estimation of gradients is costly, at the moment, and does not seem to pay off due to insufficient precision. This point also crucially depends on the accuracy of differential operator estimations.

Finally, although current error mitigation methods may improve the accuracy of cost function estimation, it is insufficient to achieve overall good convergence. We note that there is a certain degree of freedom concerning the choice of tests to present: one could present much more accurate results for 2 or 3 qubits. Nonetheless, within the context of NISQ-viability and considering the modern standards in classical numerical simulation, we believe the presented results are sufficient to cast serious doubt about the near-term applicability of quantum computers to PDEs.

ACKNOWLEDGEMENTS

This work was supported by the project AnQuC-3 of the Competence Center Quantum Computing Rhineland-Palatinate (Germany). We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

DATA AVAILABILITY

The main tests were performed on ibmq_ehningen (processor type Falcon r5.11, version 3.1.21) and ibmq_montreal (processor type Falcon r4, version 1.11.26). The coupling maps are displayed in Fig. 18. The calibration data is detailed in Table I and II. The Qiskit code can be found in https://github.com/MazenAli/VQA_Poisson1D.

[1] O. Iliev and V. Laptev, On numerical simulation of flow through oil filters, Computing and Visualization in Science 6, 139 (2004).
[2] H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keeml, F. Kuzikalla, M. Lee, C. Madonna, M. Marsh, T. Mukerji, E. H. Saenger, R. Sain, N. Saxena, S. Ricker, A. Wiegmann, and X. Zhan, Digital rock physics benchmarks—part ii: Computing effective properties, Computers & Geosciences 50, 33 (2013), benchmark problems, datasets and methodologies for the computational geosciences.
[3] M. Kabel, T. Böhlke, and M. Schneider, Efficient fixed point and newton–krylov solvers for fft-based homogenization of elasticity at large deformations, Computational Mechanics 54, 1497 (2014).
Figure 7: VQLS training on simulators with shot noise and ibmq_ehningen, BFGS optimizer with default hyperparameters, linear alternating $R_y$-$CZ$ ansatz, 5 qubits and 3 layers, RHS $\langle \text{HnX} \rangle$, Hadamard test for inner product estimation and Liu21 for $A$ decomposition. (a) Fidelity vs. number of cost function evaluations (gradient evaluations not displayed) on a simulator, training based on $C_N$. Best out of 15 initial values. (b) Fidelity vs. number of cost function evaluations (gradient evaluations not displayed) on a simulator, training based on $C_{NN}$. Best out of 15 initial values. (c) Fidelity vs. number of cost function evaluations (gradient evaluations not displayed) on ibmq_ehningen, training based on $C_{NN}$. Optimized noise-aware transpilation with SABRE and mapomatic, see also Section IVB, MEM applied, 10000 shots per circuit. Best out of 5 initial values.

[4] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103, 150502 (2009).
[5] A. Montanaro and S. Pallister, Quantum algorithms and the finite element method, Phys. Rev. A 93, 032324 (2016).
[6] K. K. Saha, W. Robson, C. Howington, I.-S. Suh, Z. Wang, and J. Nabrzyski, Advancing Algorithm to Scale and Accurately Solve Quantum Poisson Equation on Near-term Quantum Hardware, arXiv e-prints , arXiv:2210.16668 (2022), arXiv:2210.16668 [quant-ph].
[7] W. Robson, K. K. Saha, C. Howington, I.-S. Suh, and J. Nabrzyski, Advanced Quantum Poisson Solver in the NISQ era, arXiv e-prints , arXiv:2209.09366 (2022), arXiv:2209.09366 [quant-ph].
[8] A. C. Vazquez, R. Hiptmair, and S. Woerner, Enhancing the quantum linear systems algorithm using richardson extrapolation, ACM Transactions on Quantum Computing 3, 10.1145/3490631 (2022).
[9] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nature Reviews Physics 3, 625 (2021).
[10] Y. Sato, R. Kondo, S. Koide, H. Takamatsu, and N. Imoto, Variational quantum algorithm based on the minimum potential energy for solving the poisson equation, Phys. Rev. A 104, 052409 (2021).
[11] H.-L. Liu, Y.-S. Wu, L.-C. Wan, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, Variational quantum algorithm for the poisson equation, Phys. Rev. A 104, 022418 (2021).
[12] R. Demirdjian, D. Gunlycke, C. A. Reynolds, J. D. Doyle, and S. Tafur, Variational quantum solutions to the advection–diffusion equation for applications in fluid dynamics, Quantum Information Processing 21, 322 (2022).
[13] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, Variational Quantum Linear Solver, arXiv e-prints , arXiv:1909.05820 (2019).
Figure 8: VQLS training on simulators with shot noise and ibmq_ehningen, SPSA optimizer with learning rate 1, perturbation 0.1, blocking and trust region set, linear alternating $R_Y$-$CZ$ ansatz, 5 qubits and 3 layers, RHS $[\text{HnX}]$, Hadamard test for inner product estimation and Liu21 for $A$ decomposition. (a) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_N$. Best out of 15 initial values. (b) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_{NN}$. Best out of 15 initial values. (c) Fidelity vs. number of cost function evaluations on ibmq_ehningen, training based on $C_{NN}$. Optimized noise-aware transpilation with SABRE, no MEM applied, 1000 shots per circuit. Best out of 5 initial values.
Figure 9: VQLS training on simulators with shot noise and ibmq_ehningen, POWELL optimizer with default hyperparameters, linear alternating $R_Y$-$CZ$ ansatz, 5 qubits and 3 layers, RHS (HnX), Hadamard test for inner product estimation and Liu21 for $A$ decomposition. (a) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_N$. Best out of 15 initial values. (b) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_{NN}$. Best out of 15 initial values. (c) Fidelity vs. number of cost function evaluations on ibmq_ehningen, training based on $C_{NN}$. Optimized noise-aware transpilation with SABRE, no MEM applied, 10000 shots per circuit. Best out of 5 initial values.

[24] M. Cerezo, K. Sharma, A. Arrasmith, and P. J. Coles, Variational quantum state eigensolver, npj Quantum Information 8, 113 (2022).
[25] C. G. Broyden, The Convergence of a Class of Double-rank Minimization Algorithms: 2. The New Algorithm, IMA Journal of Applied Mathematics 6, 222 (1970). [https://academic.oup.com/imamat/article-pdf/6/3/222/1848059/6-3-222.pdf](https://academic.oup.com/imamat/article-pdf/6/3/222/1848059/6-3-222.pdf).
[26] R. Fletcher, A new approach to variable metric algorithms, The Computer Journal 13, 317 (1970). [https://academic.oup.com/comjnl/article-pdf/13/3/317/988678/130317.pdf](https://academic.oup.com/comjnl/article-pdf/13/3/317/988678/130317.pdf).
[27] D. Goldfarb, A family of variable-metric methods derived by variational means, Mathematics of Computation 24, 23 (1970).
[28] D. F. Shanno, Conditioning of quasi-newton methods for function minimization, Mathematics of Computation 24, 647 (1970).
[29] A. Pellow-Jarman, I. Sinayskiy, A. Pillay, and F. Petruccione, A comparison of various classical optimizers for a variational quantum linear solver, Quantum Information Processing 20, 202 (2021).
[30] Recently, IBM extended their software tools by adding runtime primitives with streamlined error mitigation. Some noteworthy error mitigation techniques not considered in our tests are zero noise extrapolation and probabilistic error cancellation [19, 20].
[31] G. Li, Y. Ding, and Y. Xie, Tackling the qubit mapping problem for nisq-era quantum devices, in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS ’19 (Association for Computing Machinery, New York, NY, USA, 2019) p. 1001–1014.
[32] P. D. Nation, H. Kang, N. Sundaresan, and J. M. Gambetta, Scalable mitigation of measurement errors on quantum computers, PRX Quantum 2, 040326 (2021).
[33] P. D. Nation and M. Treinish, Suppressing quantum circuit errors due to system variability, arXiv e-prints, arXiv:2209.15512 (2022), arXiv:2209.15512 [quant-ph].
Figure 10: VQLS training on simulators with shot noise and ibmq_ehningen, NFT optimizer with reset interval 9, linear alternating $R_Y$-$CZ$ ansatz, 5 qubits and 3 layers, RHS [HinX]. Hadamard test for inner product estimation and Liu21 for $A$ decomposition. (a) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_N$. Best out of 15 initial values. (b) Fidelity vs. number of cost function evaluations on a simulator, training based on $C_{NN}$. Best out of 15 initial values. (c) Fidelity vs. number of cost function evaluations on ibmq_ehningen. “No EM” means only optimized transpilation with SABRE and “EM” means additionally noise-aware transpilation with mapomatic and MEM. Best out of 5 initial values.

Figure 11: Trace distance (y-axis) vs. fidelity (x-axis). The last point on the x-axis is 0.99999999.

[34] G. Ravi, K. N. Smith, P. Gokhale, A. Mari, N. Earnest, A. Javadi-Abhari, and F. T. Chong, Vaqem: A variational approach to quantum error mitigation, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE Computer Society, Los Alamitos, CA, USA, 2022) pp. 288–303.

[35] K. M. Nakanishi, K. Fujii, and S. Todo, Sequential minimal optimization for quantum-classical hybrid algorithms, *Phys. Rev. Research* 2, 043158 (2020).

[36] M. Oliv, A. Matic, T. Messerer, and J. M. Lorenz, Evaluating the impact of noise on the performance of the Variational Quantum Eigensolver, arXiv e-prints, arXiv:2209.12803 (2022), arXiv:2209.12803 [quant-ph].
[37] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, *Nature* **549**, 242 (2017).

[38] J. Spall, Accelerated second-order stochastic optimization using only function measurements, in *Proceedings of the 36th IEEE Conference on Decision and Control*, Vol. 2 (1997) pp. 1417–1424 vol.2.

[39] M. J. D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, *The Computer Journal* **7**, 155 (1964). [https://academic.oup.com/comjnl/article-pdf/7/2/155/959784/070155.pdf](https://academic.oup.com/comjnl/article-pdf/7/2/155/959784/070155.pdf)

[40] G. Iannelli and K. Jansen, Noisy Bayesian optimization for variational quantum eigensolvers, PoS **LATICE2021**, 251 (2022) [arXiv:2112.00426 [quant-ph]]

[41] J. Gacon, C. Zoufal, G. Carleo, and S. Woerner, Simultaneous Perturbation Stochastic Approximation of the Quantum Fisher Information, *Quantum* **5**, 567 (2021)

[42] S. H. Sack, R. A. Medina, A. A. Michailidis, R. Kueng, and M. Serbyn, Avoiding barren plateaus using classical shadows, *PRX Quantum* **3**, 020365 (2022)
(a) Linear $R_y$-$CZ$, 5 qubits, 2 layers.

(b) Linear alternating $R_y$-$CZ$, 5 qubits, 2 layers.

(c) Linear alternating periodic $U3$-$CX$, 5 qubits, 2 layers.

(d) Linear alternating periodic bidirectional $R_z$-$CX$, 5 qubits, 2 layers.

(e) QAOA (inspired) ansatz, 5 qubits, 2 layers.

(f) QAOA periodic, 5 qubits, 2 layers.

Figure 12: Ansätze used for $|\psi(\theta)\rangle$ in this work.
Figure 13: Circuits for estimating inner products with $|\psi(\theta)\rangle = U(\theta) |0\rangle$ and $|f\rangle = U_f |0\rangle$. 

Figure 14: Circuit for $I^{{\otimes n-1}} \otimes X$ required for all decompositions.

Figure 15: Sato21: circuit for estimating expectation of $A$ as in [10].
Figure 16: Liu21: circuits for estimating expectation of $A$ as in [11].

Figure 17: Liu21Grouped: circuit for rotating into the common eigenbasis of the commuting operators proposed in [11].
Figure 18: IBM backend connectivity maps ibmq_ehningen (a) and ibmq_montreal (b). Lighter nodes/connections means a larger error.
| Entry | Qubit | X error | Y error | R error | Gate time (ns) | State time (ns) |
|-------|-------|---------|---------|---------|---------------|----------------|
| 0_1   | 0.005152644673167811 | -0.3400425781524352 | 0.010800000000000032 | 5.126925983720086 | 846.2222222222222 | 15_12:0.01030221725514388 |
| 0_2   | 0.0006089056917463167 | 5.268148656305129 | 0.009 | 846.2222222222222 | 15_12:0.01030221725514388 |
| 0_3   | 0.0003881828478182781 | 12_15:0.01030221725514388 | 0.00022478378392899726 | 10_7:320; 10_12:259.55555555555554 | 12_15:0.01030221725514388 |
| 0_4   | 0.00019174605488026967 | 12_10:0.00601257299190297 | 0.0002940226149296246 | 12_15:0.01030221725514388 |
| 0_5   | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_6   | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_7   | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_8   | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_9   | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_10  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_11  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_12  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_13  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_14  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_15  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_16  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_17  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_18  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |
| 0_19  | 0.00019174605488026967 | 12_13:0.006548795528996093 | 0.00015587488620965434 | 12_15:0.01030221725514388 |

Table I: Calibration data ibmq_ehningen.
| Level | Frequency (GHz) | Readout length (ns) | Gate time (ns) |