Title

Classical Optimizers for Noisy Intermediate-Scale Quantum Devices

Authors

Lavrijsen, Wim

Tudor, Ana

Müller, Juliane

et al.

Publication Date

2020-10-16

DOI

10.1109/qce49297.2020.00041

Peer reviewed

Classical Optimizers for Noisy Intermediate-Scale Quantum Devices

Wim Lavrijsen, Ana Tudor, Juliane Müller, Costin Iancu, Wibe de Jong
Lawrence Berkeley National Laboratory
{wlavrijsen,julianemueller,cciancu,wadejong}@lbl.gov, anamtudor@berkeley.edu

ABSTRACT

We present a collection of optimizers tuned for usage on Noisy Intermediate-Scale Quantum (NISQ) devices. Optimizers have a range of applications in quantum computing, including the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization (QAOA) algorithms. They have further uses in calibration, hyperparameter tuning, machine learning, etc. We employ the VQE algorithm as a case study. VQE is a hybrid algorithm, with a classical minimizer step driving the next evaluation on the quantum processor. While most results to date concentrated on tuning the quantum VQE circuit, our study indicates that in the presence of quantum noise the classical minimizer step is a weak link and a careful choice combined with tuning is required for correct results. We explore state-of-the-art gradient-free optimizers capable of handling noisy, black-box, cost functions and stress-test them using a quantum circuit simulation environment with noise injection capabilities on individual gates. Our results indicate that specifically tuned optimizers are crucial to obtaining valid science results on NISQ hardware, as well as projecting forward on fault tolerant circuits.

1 INTRODUCTION

Hybrid quantum-classical algorithms are promising candidates to exploit the potential advantages of quantum computing over classical computing on current quantum hardware. Target application domains include the computation of physical and chemical properties of atoms and molecules [10], as well as optimization problems [9, 34] such as graph MaxCut.

These hybrid algorithms execute a classical optimizer that iteratively queries a quantum algorithm that evaluates the optimization objective. An example is the Variational Quantum Eigensolver (VQE) algorithm [20] applied in chemistry, where the objective function calculates the expectation value of a Hamiltonian $\mathcal{H}$ given an input configuration of a simulated physical system. The Hamiltonian describes the energy evolution of the system, thus the global minimum represents the ground level energy. The classical side variationally changes the parametrized input configuration until convergence is reached, thereby finding the eigenvalue and eigenstate of the ground energy of $\mathcal{H}$. Quantum Approximate Optimization Algorithms (QAOA) [9, 34] employ a similar approach.

For the foreseeable future, quantum algorithms will have to run on Noisy Intermediate-Scale Quantum (NISQ) devices which are characterized by a small number of noisy, uncorrected qubits. Hybrid methods are considered auspicious on such devices due to:

(1) the expectation that their iterative nature makes them robust to noise; and
(2) reduced chip coherence time requirements because of the single Hamiltonian evaluation per circuit execution.

However, these considerations relate to the quantum side of the hybrid approach. Rather, as we will show in this paper, the impact of noise on *both* the classical and quantum parts needs to be taken into account. In particular, the performance and mathematical guarantees, regarding convergence and optimality in the number of iterations, of commonly used classical optimizers rest on premises that are broken by the existence of noise in the objective function. Consequently, they may converge too early, not finding the global minimum, get stuck in a noise-induced local minimum, or even fail to converge at all.

For chemistry, the necessity of developing robust classical optimizers for VQE in the presence of hardware noise has already been recognized [20]. However, the first published hardware studies side-stepped optimizers by performing a full phase space exploration [8, 18, 31] and backfitting the solution to zero noise. This works for low qubit count and few minimization parameters, but is not tractable at the $\mathcal{O}(100)$ qubit concurrency soon expected on NISQ-era devices, nor for the number of parameters needed for realistic problems. To our knowledge, QAOA studies also ignore the effects of the noise on the classical optimizers.

In this study, we want to understand the requirements on classical optimizers for hybrid algorithms running on NISQ hardware and which optimization methods best fulfill them. We use VQE as the testing vehicle, but expect the findings to be readily applicable to QAOA and other hybrid quantum-classical methods which employ similar numerical optimization. The goals and contributions of our empirical study are twofold:

- A practical software suite of classical optimizers, directly usable from Python-based quantum software stacks, together with a tuning guide. We consider factors such as the quality of the initial solution and availability of bounds, and we test problems with increasing number of parameters to understand scalability of the selected methods.
- A study of the optimizers' sensitivity to different types of noise, together with an analysis of the impact on the full VQE algorithm. We consider the domain science perspective: some level of experimental error is expected and acceptable, as long as the result is accurate and the errors can be estimated. We run simulations at different noise levels and scale, for several science problems with different optimization surfaces, finding the breaking points of the minimizers and the algorithm for each.

We have taken a very practical tack and first evaluated the minimizers from SciPy [29]. These include methods such as the quasi-Newton BFGS [22] algorithm, and are the default choice of many practitioners. Most optimization tools in standard Python and MATLAB software are not noise-aware and, as we have found in our evaluations, actually fail in the presence of quantum noise. Some optimizers are more robust due to the smoothing effect of the underlying methods used (e.g. modeling in trust region methods), but that is seldom by design.

Fortunately, applied mathematicians in the optimization community have long been working on this type of problem and have

provided high quality, open source, software. Based on their recommendation, our final selection contains representative methods of (hybrid) mesh (ImFil [16], NOMAD [17]); local fit (SnobFit [15]); and trust regions (PyBobyqa [5, 6]). Python and C++ are far more widely used in quantum computing than MATLAB. Thus, we have rewritten optimizers where necessary from MATLAB into Python, while ensuring, through a suite of unit tests, reproducible deterministic behavior after porting, and provided consistent interfaces and plugins for high level quantum frameworks such as Qiskit [1] and Cirq [12]. These products have been packaged into SCIKIT-QUANT [28]. The optimization package in SCIKIT-QUANT also provides tutorial notebooks with tips and hints for hyper-parameter optimization, and an evaluation harness to quickly assess applicability to new problems.

SCIKIT-QUANT has been evaluated on three VQE problems (ethylene C_2H_6 rotation and bond stretching, and Hubbard model simulation), each with different optimization requirements. The results indicate that a *suite* of minimizers is needed to match specific strengths to specific problems. Achieving high quality solutions is aided by domain science information, if available, such as a good initial parameters, knowledge of local minima, or the need to search around inaccessible regions. Such information is problem specific and in practice we observe different performance benefits with different optimizers from its inclusion. Where this information is *not* available, our study indicates that the best results are obtained by composing local and global optimizers, leveraging their respective strengths, during the VQE algorithm run.

The organization of this paper is as follows. In Section 2, we give a brief background on numerical optimization and our requirements on optimizers. In Section 3 we describe the optimizers available in SCIKIT-QUANT in more detail. We provide the necessary background on hybrid quantum-classical algorithms in Section 4 and we describe the impact of noise in Section 5. Our numerical experiments are presented in Section 6 and discussed in Section 7. We compare our work with related studies in Section 8 and finally summarize the main conclusions in Section 9.

2 NUMERICAL OPTIMIZATION

In variational hybrid quantum-classical algorithms, such as VQE, the execution on the quantum processor evaluates the objective function to be optimized classically. In most cases, it is not possible to calculate gradients directly, thus derivative-free optimization methods are required. For a deterministic function $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ over a domain Ω of interest that has lower and upper bounds on the problem variables, derivative-free algorithms require only evaluations of f but no derivative information. They assume that the derivatives of f are neither symbolically nor numerically available, and that bounds, such as Lipschitz constants, for the derivatives of f are also unavailable.

Optimizers are judged on the quality of the solution and on their speed and scalability. A good solution has a short distance to the true global optimum, high accuracy of the optimal parameters found, or both. A good overview and thorough evaluation of derivative-free algorithms can be found in Rios et al. [27]. The main criteria for matching an optimizer to a problem are the convexity and the smoothness of the optimization surfaces. Convexity has the familiar meaning; smoothness in our context requires that the function is "sufficiently often differentiable". In VQE, the shape of the optimization surface is determined by the ansatz, and although typical surfaces are smooth, noise can change this considerably.

Figure 1 shows the evolution of the optimization surface for a single parameter in a simple VQE problem (rotation/torsion of an ethylene molecule; 4 qubits, 2 parameters) for increasing levels of Gaussian gate noise (detailed background on this and other studies is provided in Sections 4 and 5). For low noise, the optimization surface is convex around the global minimum and smooth. For increasing levels of noise, the optimization surface becomes both non-convex and non-smooth. It gets substantially worse for more complex problems: because circuit depth increases, because the number of parameters increases the likelihood of noise-induced local minima, and because entanglement over many qubits means that the effects of gate noise become non-local. This can be seen in Figure 2, which displays the effect of noise on an 8 qubit Hubbard model simulation, with 14 parameters at a moderate level of gate noise of $\sigma = 0.01$rad. (cf. the mid-range in the ethylene figure). We are thus interested in optimizers that perform well across the whole range of behaviors: convex and non-convex surfaces, smooth and non-smooth surfaces.

2.1 Optimizer Selection Criteria

The criteria for selecting optimizers that we considered are:

(1) Ability to find a good solution in the presence of noise, potentially using different methods for different types of surfaces.
(2) Scalability with the number of parameters, as this determines the asymptotic behavior on future quantum hardware that allows the simulation of larger problems.
(3) Number of samples (queries to the objective function) required and precision needed, which affects scaling and wall-clock time spent on the quantum chip.
(4) Implementation performance and ability to parallelize, as these affect scaling and wall-clock time spent on the classical side.

There are two common strategies for optimizing noisy outcomes: optimize for the expected value of the response, or for the worst case [25]. Quantum simulations, being probabilistic in nature, fit the former: many runs ("shots") of a circuit are required to obtain the output distribution, which is then inclusively averaged over local noise sources.

2.2 Baseline Optimizers

Under the assumption that the objective function is still continuously differentiable, quasi-Newton methods can be used. These approximate the first (and often the second) derivative from the evaluations at different points. Such methods work better if a detailed understanding of the noise is available, allowing selection of good step sizes and properly weigh evaluations when incorporating them into the approximation of the derivatives. In the case of BFGS, which has been used by VQE developers for algorithm development on quantum simulators[1], each new evaluation is instead added to the current derivative estimate with equal weight to all points collected so far combined. This means that BFGS is easily thrown off when function values are noisy.

[1] As opposed to real hardware.

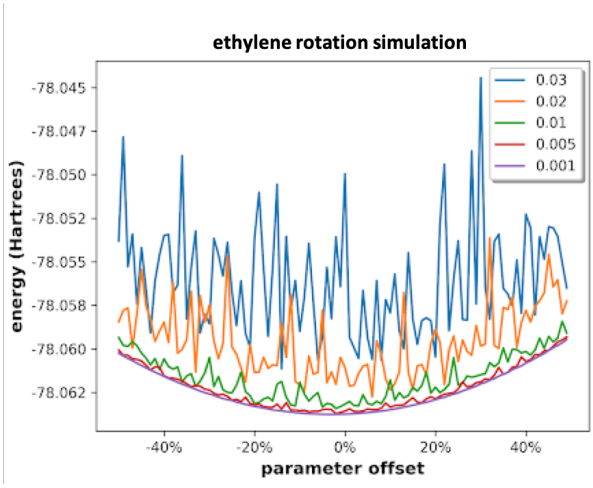

Figure 1: *Evolution of the optimization surface in the main parameter for ethylene rotation simulation as a function of Gaussian gate noise. The surface goes from convex and smooth to non-convex, non-smooth as noise increases.*

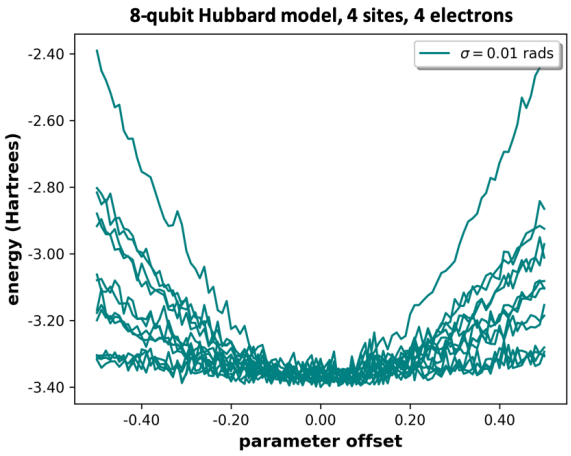

Figure 2: *Optimization surfaces of all 14 parameters with Gaussian gate noise of* $\sigma = 0.01$rad. *in a Hubbard model simulation of 4 sites with 4 electrons (see Section 6 for full details).*

Given that it is still a common first choice, we retain BFGS as a baseline for comparisons for our initial experiments and candidate optimizer selection for SCIKIT-QUANT. We use the SciPy [29] BFGS implementation and tune it for all input problems. We have also evaluated a range of other methods for which implementations are readily available in Python, such as the Nelder-Mead simplex method [11] (considered by McClean et al. [20] in their initial VQE analysis paper), RBFOpt [7], Cobyla [24], DYCORS[26], and CMA-ES [13, 14]. These methods do not make the hard assumptions about data quality that BFGS does, leaving them somewhat more robust to noise. Based on our evaluation, we find Cobyla to outperform and thus we use it as a second baseline for subsequent comparisons.

3 SCIKIT-QUANT OPTIMIZERS

The initial selection of optimizers packaged in SCIKIT-QUANT consists of NOMAD, ImFil, SnobFit, and BOBYQA; each detailed in the rest of this section. This choice is motivated by the evaluation of Rios et al. [27] combined with open-source availability and ease of porting[2] to Python. Rios et al. [27] indicate the following trends:

- In terms of scalability, SnobFit and NOMAD may have scalability challenges with the number of parameters (tested up to 300). ImFil and BOBYQA are among the fastest optimizers.
- For convex optimization surfaces, BOBYQA and SnobFit perform well for smooth surfaces, while NOMAD and ImFil perform better for non-smooth surfaces.
- For non-convex optimization surfaces, SnobFit and NOMAD are good for smooth surfaces, while ImFil and NOMAD are good for non-smooth surfaces.

In the rest of this section we give a short description of each algorithm together with their tunable knobs that affect their performance and solution quality. As common characteristics we note that all derivative-free optimizers employ sampling strategies and require a minimum number of samples to get started. This allows a common interface to employ parallelization of the quantum step, even if the original codes do not support this directly. Sampling requires that the parameter space is bounded, or that search vectors are provided. Most optimizers can make use of further detailed science domain information, such as the magnitude and shape of uncertainties, local functional descriptions, inaccessible regions, etc. If no such information is provided or available, they will choose reasonable defaults, e.g. assumption of homogeneous, symmetric, uncertainties; and qubic or quadratic local functional behavior on a small enough region. Inaccessible regions can simply be communicated by returning NaN from the objective function.

3.1 NOMAD

NOMAD, or *Nonlinear Optimization by Mesh Adaptive Direct Search (MADS)* [17] is a C++ implementation of the MADS algorithm [2–4]. MADS searches the parameter space by iteratively generating a new sample point from a mesh that is adaptively adjusted based on the progress of the search. If the newly selected sample point does not improve the current best point, the mesh is refined. NOMAD uses two steps (*search* and *poll*) alternately until some preset stopping criterion (such as minimum mesh size, maximum number of failed consecutive trials, or maximum number of steps) is met. The search step can return any point on the current mesh, and therefore offers no convergence guarantees. If the search step fails to find an improved solution, the poll step is used to explore the neighborhood of the current best solution. The poll step is central to the convergence analysis of NOMAD, and therefore any hyperparameter optimization or other tuning to make progress should focus on the poll step. Options include: poll direction type (local model, random, uniform angles, etc.), poll size, and number of polling points.

The use of meshes means that the number of evaluations needed scales at least geometrically with the number of parameters to be optimized. It is therefore important to restrict the search space

[2]Note that while we ported the same algorithms, they evaluated different implementations, which may affect some of the total running time.

as much as possible using bounds and, if the science of the problem so indicates, give preference to polling directions of the more important parameters.

In SCIKIT-QUANT we incorporate the published open-source NOMAD code through a modified Python interface.

3.2 ImFil

Implicit Filtering (ImFil [16]) is an algorithm designed for problems with local minima caused by high-frequency, low-amplitude noise and with an underlying large scale structure that is easily optimized. ImFil uses difference gradients during the search and can be considered as an extension of coordinate search. In ImFil, the optimization is controlled by evaluating the objective function at a cluster (or stencil) of points within the given bounds. The minimum of those evaluations then drives the next cluster of points, using first-order interpolation to estimate the derivative, and aided by user-provided exploration directions, if any. Convergence is reached if the "budget" for objective function evaluations is spent, if the smallest cluster size has been reached, or if incremental improvement drops below a preset threshold.

The initial clusters of points are almost completely determined by the problem boundaries, making ImFil relatively insensitive to the initial solution and allows it to easily escape from local minima. Conversely, this means that if the initial point is known to be of high quality, ImFil must be provided with tight bounds around this point, or it will unnecessarily evaluate points in regions that do not contain the global minimum.

As a practical matter, for the noisy objective functions we studied, we find that the total number of evaluations is driven almost completely by the requested step sizes between successive clusters, rather than finding convergence explicitly.

For SCIKIT-QUANT we have rewritten in Python the original ImFil MATLAB implementation available.

3.3 SnobFit

Stable Noisy Optimization by Branch and FIT (SnobFit) [15] is an optimizer developed specifically for optimization problems with noisy and expensive to compute objective functions. SnobFit iteratively selects a set of new evaluation points such that a balance between global and local search is achieved, and thus the algorithm can escape from local optima. Each call to SnobFit requires the input of a set of evaluation points and their corresponding function values and SnobFit returns a new set of points to be evaluated, which is used as input for the next call of SnobFit. Therefore, in a single optimization, SnobFit is called several times. The initial set of points is provided by the user and should contain as many expertly chosen points as possible (if too few are given, the choice is a uniformly random set of points, and thus providing good bounds becomes important). In addition to these points, the user can also specify the uncertainties associated with each function value. We have not exploited this feature in our test cases, because although we know the actual noise values from the simulation, properly estimating whole-circuit systematic errors from real hardware is an open problem.

As the name implies, SnobFit uses a branching algorithm that recursively subdivides the search space into smaller subregions from which evaluation points are chosen. In order to search locally, SnobFit builds a local quadratic model around the current best point and minimizes it to select one new evaluation point. Other local search points are chosen as approximate minimizers within a trust region defined by safeguarded nearest neighbors. Finally, SnobFit also generates points in unexplored regions of the parameter space and this represents the more global search aspect.

For SCIKIT-QUANT we have rewritten in Python the original SnobFit MATLAB implementation available.

3.4 BOBYQA

BOBYQA (Bound Optimization BY Quadratic Approximation) [23] has been designed to minimize bound constrained black-box optimization problems. BOBYQA employs a trust region method and builds a quadratic approximation in each iteration that is based on a set of automatically chosen and adjusted interpolation points. New sample points are iteratively created by either a "trust region" or an "alternative iterations" step. In both methods, a vector (step) is chosen and added to the current iterate to obtain the new point. In the trust region step, the vector is determined such that it minimizes the quadratic model around the current iterate and lies within the trust region. It is also ensured that the new point (the sum of the vector and the current iterate) lies within the parameter upper and lower bounds. BOBYQA uses the alternative iteration step whenever the norm of the vector is too small, and would therefore reduce the accuracy of the quadratic model. In that case, the vector is chosen such that good linear independence of the interpolation points is obtained. The current best point is updated with the new point if the new function value is better than the current best function value. Note that there are some restrictions for the choice of the initial point due to the requirements for constructing the quadratic model. BOBYQA may thus adjust the initial automatically if needed.

Although it is not intuitively obvious that BOBYQA would work well on noisy problems, we find that it performs well in practice if the initial parameters are quite close to optimal and the minimum and maximum sizes of the trust region are properly set. This is rather straightforward to do for the specific case of VQE, where a good initial guess can be obtained relatively cheaply from classical simulation. For Hubbard model problems, which have many (shallow) local minima, BOBYQA does not perform nearly as well.

In SKIKIT-QUANT, we use the existing PyBobyqa implementation [5, 6] directly.

3.5 Validation and Tuning

We have validated the SCIKIT-QUANT implementations for correctness and performance using a suite of unit tests. For ImFil and SnobFit, which have been ported from MATLAB, we have thoroughly tested correctness, using their original tests as well as our own. For NOMAD and PyBobyqa we invoke the original implementations, limiting the need for testing beyond the application programming interface. All tests have been included in the SCIKIT-QUANT repository.

We have chosen defaults for each optimizer that should work best for the type of optimization surfaces and noise behavior observed in the problems considered. Several of these choices are different from the original defaults, and in all cases involved at

least an increase of the number of samples per iteration (BOBYQA and NOMAD in particular benefit here) or a tightening of the convergence criteria (important for SnobFit). This trades wall clock performance with science performance. In the case of ImFil, a functional change was needed: without a reduction in the smallest step scales, chemical accuracy could not be achieved. We balanced this cost with a reduction in the allowed number of internal iterations in the interpolation on a stencil.

We consider good default values extremely important: as a practical matter, domain scientists tend to judge optimizers based on trial runs on their problem at hand, rather than first studying their problem's mathematical properties and only then searching for an optimizer to match, with different tuning as needed. That (faulty) approach may well cause them to miss out on the best choice. Good domain-specific defaults ameliorate this practical issue somewhat.

4 HYBRID QUANTUM-CLASSICAL ALGORITHMS

The hybrid quantum-classical algorithms we consider iteratively alternate between a classical numerical optimizer and a quantum algorithm that evaluates some objective to be minimized. The classical optimizer varies a set of parameters that determine the input state for the quantum processor to prepare. The quantum side then executes an algorithm resulting in measurement and some output distribution of probabilities. This distribution is mapped into an objective function value that the classical optimizer can handle, such as a single floating point number, e.g., one representing the expected energy of a physical system (see Figure 3).

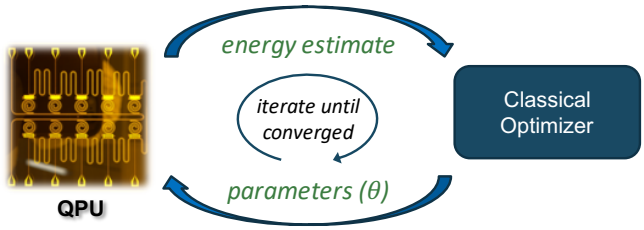

Figure 3: *VQE algorithm schematic. The goal of the algorithm is to find* $E_0(\theta) = \min_\theta(\langle\psi(\theta)|\mathcal{H}|\psi(\theta)\rangle/\langle\psi(\theta)\psi(\theta)\rangle)$ *with the classical optimizer changing the input by varying optimization parameters* θ *and the quantum chip calculating the expectation value of* $\mathcal{H}$.

In the Variational Quantum Eigensolver approach for solving chemistry and physics problems, the objective function calculates the expectation value of the Hamiltonian $\mathcal{H}$ associated with a configuration of the simulated physical system. Without noise, the optimization surface is expected to be smooth and convex around the global minimum. Bounds and constraints to help the optimizer and analysis are often straightforward to obtain from physical laws, e.g. there should be no loss of particles.

In Quantum Approximate Optimization Algorithms, the state is prepared by a p-level circuit specified by $2p$ variational parameters. Even at the lowest circuit depth (p=1), QAOA has non-trivial provable performance guarantees. Initial QAOA exemplars have been selected from the domain of graph optimization problems such as MaxCut. The optimization surfaces generated by QAOA problems can be arbitrarily complex and bounds and constraints are harder to define as they need not be physical.

Because of these last differences, understanding the impact of noise on the behavior of hybrid algorithms is more straightforward for VQE and we will concentrate our study on its behavior. However, since we do not restrict the study to realistic noise levels only, but push the optimizers to their breaking point, we believe that our findings are directly applicable to the higher complexity in QAOA algorithms as well. For more details, see Section 8.

4.1 Role of the Ansatz in VQE

The classical optimizer is not free to choose input states for VQE, but constrained by a parametrized *ansatz*, which describes the range of valid physical systems and thus determines the optimization surface. A good ansatz provides a balance between a simple representation (and thus simple operators in the quantum circuit), efficient use of available native hardware gates, and sufficient sensitivity of the objective with the input parameters. An effective ansatz can greatly reduce circuit depth, search space, and the number of steps necessary to convergence.

For now, ansatz design is still an art that requires detailed insights from the domain science to uncover symmetries and to decide which simplifications are acceptable. However, our main interest is to push the optimizers. Since a better ansatz will simply allow the domain scientist to work on larger, more complex, problems that equally push the optimizer harder, we will restrict ourselves to the commonly used, and practical, *unitary coupled cluster* ansatz (UCC ansatz) for all studies. For physical systems, the UCC ansatz can be thought of as describing the movements of individual particles (linear terms) and those of interacting (e.g. through electric charge) pairs of particles (quadratic terms). It is simple to map and, because particles such as electrons are indistinguishable, easy to find symmetries to reduce the number of parameters needed to describe all valid configurations.

Besides the number of parameters, the choice of ansatz also affects the number of qubits used. For example, the UCC ansatz provides for simple physical interpretations, such as '1' meaning that a site or orbital is occupied by an electron, and '0' meaning that it is unoccupied. Add a second qubit for spin up and down, and two qubits can fully describe a site or orbital.[3] However, there is a clear inefficiency here: it is unnecessary to describe the spin of an unoccupied site. But changing to a more compact representation requires changing the ansatz and the operators, which can actually make the problem harder to solve. Published results [8, 18, 31] comprise only two and four qubit experiments with two parameters. In our studies we have used 4 and 8 qubit problems, with the number of parameters ranging from 2 to 14.

4.2 VQE Quantum Processor Step

The quantum circuit consists of two parts: a state preparation and an evolution. The state preparation takes the chip from its computational ground state to the intended initial state as set by the classical

[3] It is still completely up to the domain scientist to determine which and thus how many sites are relevant for the problem they are trying to solve, which is the most important driver of the number of qubits needed.

optimizer. The evolution works by computing successive steps in "imaginary time" ($e^{-i\mathcal{H}\tau}$ with $\tau = it$). This process attenuates the contributions of the eigenvectors of the Hamiltonian proportional to the exponent of their respective eigenvalues. Thus, after a sufficient number of steps, only the component of the smallest eigenvalue is left. The chip readout is then a probability distribution of bit strings that represents the estimated ground energy eigenstate, from which the estimated energy is then calculated classically using the Hamiltonian. The mapping of the measured probability distribution to a single number (the energy) is non-linear because the input is constrained to be physical and sum to 1. It is thus not possible to make any general inference about the uncertainty distribution of the estimated energy from the expected errors in the probability distribution, but only about specific problem instances.

5 IMPACT OF NOISE

VQE is considered to have some robustness against noise due to its iterative nature and hence is expected to be well suited for upcoming NISQ devices. Nevertheless, the need for studying the dynamics of the full hybrid VQE algorithm has been identified early on [20] as a prerequisite for successfully running it on NISQ hardware.

There are two components to this problem: 1) understanding how well optimizers handle noisy data; and 2) understanding how well the full quantum-classical algorithm handles noise.

5.1 Accounting for Noise Sources

There are a range of ways that noise enters the final result: from electronic noise and quantum crosstalk, to decoherence and calibration inaccuracies. How the output of a quantum circuit is affected by noise is an open research problem, with no accurate predictive models available, even when restricted to a specific chip instance. Our main concerns, however, are about overall magnitude of noise and the effects on the shape of the optimization surface.

In our study, we provide coverage of the problem domain by varying the magnitude of the noise in simulation by a wide range, and by studying different problems with a priori different optimization surfaces. The actual noise impact for a given hardware instance is likely to be captured within our parameter sweep. The upshot is that we study a wide range of noisy profiles across different optimizers to arrive at a *map and guidance* for actual experiments. The goal is explicitly not to find and describe *the* single way, if any such exists, of how VQE behaves with a given noise model, nor to find the one optimizer that should be used for all VQE problems. It is, after all, well known in the applied math community that there is no such thing as a "free lunch," meaning that each optimizer has specific strengths, none are best in all instances, and each problem needs to be individually matched to the appropriate optimizer(s).

To account for the impact of noise sources, we consider an empirical approach where we inject noise as Gaussian-distributed over-/under-rotations with an added orthogonal component onto the circuit gates. This ensures several realistic properties: noise increases with circuit depth and complexity, and two-qubit gates have larger contributions than one-qubit gates.

We do not add coherent or correlated noise sources, for the reasons explained below. The measurement result is a probability distribution of bit strings, and any stochastic noise behaves on it in a similar way: it redistributes relative counts with rates proportional to the content and with the same equilibrium in the limit, namely a uniform distribution. Coherent and correlated noise sources can, on the other hand, potentially result in *any biased distribution*, making their study meaningless, unless taken from the behavior of actual hardware. But that would, of course, limit their relevance to that specific hardware. Further, as detailed below, VQE has more "builtin" robustness against coherent than against stochastic noise. Coherent noise can also be expected to more easily produce non-physical outcomes (e.g. fewer or more particles in the final than in the input states); those measurements can be filtered out and discarded. Last but not least, orthogonal error mitigation techniques such as Randomized Compiling [33] have been shown to alleviate coherent errors by making them stochastic.

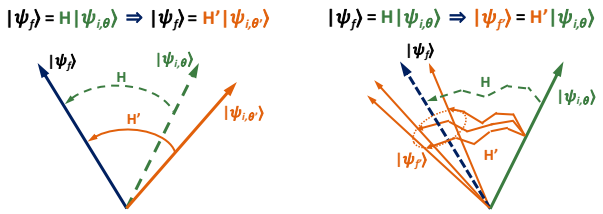

Figure 4: *Impact of noise types. The optimizer can "compensate" in the choice of input for the predictable effects of systematic/coherent noise (left) and thus still find the global minimum. But stochastic noise leads to a "random walk" away from the intended output state (right), resulting in an increasingly diminished likelihood of the objective function returning the global minimum.*

We do not factor in an additional noise contribution from measurement errors: shot noise is expected to be unbiased (i.e. it can be averaged out to zero noise in the limit by taking a large number of measurements). In other words, it affects the overall magnitude of stochastic noise sources, which we already sweep, not what we most care about: the shape changes in the optimization surface.

5.2 Interplay with Minimizer

Some general observations can be made about the different impacts of coherent and stochastic errors, and why the distinction matters on hybrid quantum-classical algorithms that involve a classical optimizer, such as VQE.

Quantum computing is very sensitive to noise, because a noisy execution is just as valid as a noise-free one: without error correction codes, there is no distinguishing between valid and erroneous states. Therefore, if a circuit is intended to simulate the evolution of some Hamiltonian $\mathcal{H}$, then a single noisy run can be seen as the evolution of some other Hamiltonian $\mathcal{H}'$. As long as the noise level is "small enough," the eigenstates[4] of $\mathcal{H}$ and $\mathcal{H}'$ will be close.

The algorithm is somewhat robust to coherent errors. By definition, changes around the output state that represents the global minimum are, to first order, zero for small linear changes in the input state. With a systematic difference between $\mathcal{H}'$ and $\mathcal{H}$, the global minimum is still found by the optimizer compensating accordingly in the input state, see Figure 4 (left). Thus, even as the calculated minimum energy may still be very close, the optimal

[4] And eigenvalues, but that is irrelevant, because these are calculated classically based on the output distribution.

parameters found are likely to be systematically off. There is a further twist here for VQE: the ansatz restricts the input states that can be chosen, thus VQE will be more quickly affected by coherent errors than hybrid algorithms in general.

The algorithm has challenges with stochastic noise. The picture changes significantly with stochastic noise: each execution of the circuit is in effect a different $\mathcal{H}'$. Once close to the global minimum, the minimizer will not be able to distinguish the outputs of runs with different inputs, as the changes get washed out in the noise (as shown earlier in Figure 2).

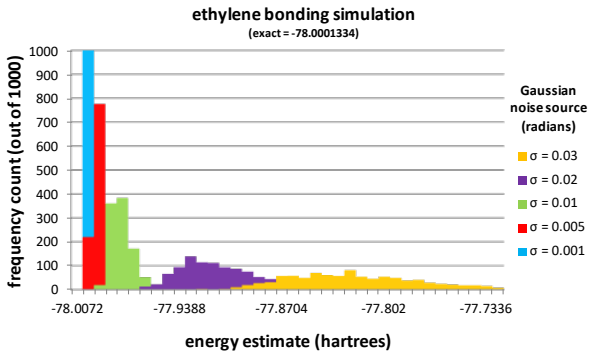

Figure 5: *Objective function evaluation at the optimal parameters for ethylene bond breaking simulation under increasing stochastic errors. Energy estimates broaden in distribution and the average lifts. Eventually, with increasing noise, the true global minimum is never returned.*

With sufficient symmetry in the optimization profile or a functional description based on the domain science, the optimizer can still find the correct optimal parameters by searching for a *robust* global minimum or doing a local fit. However, any execution at the optimal parameters will calculate an output distribution that is some random walk away from the intended state, as the errors (in particular those on the control qubit of CNOTs) do not commute with the circuit as a whole, see Figure 4 (right). When calculating the energy objective from any of these noisy outputs that are close to, but not at, the global minimum, the results will by definition be higher than the ground state energy[5]. With increasing noise, the likelihood of the true global minimum energy being returned by the objective function goes to zero, as shown in Figure 5.

6 RESULTS

As study cases, we used the C-C axis rotation and bond stretching and breaking of the ethylene (C_2H_6) molecule (see Figure 6), representing two different chemical transformation processes. In the rotation and bonding processes, the character of the wave function changes drastically. For example, in the C-C axis rotation $\Pi - \Pi$ bonds are broken/formed.

We also used a Hubbard simulation of 4 sites, occupied with either 4 or 2 electrons (see Figure 7). In the Hubbard simulations, we use a hopping term of 1.0, a Coulomb term of 2.0, and in the 4 electron case add a chemical potential of 0.25. The electrons have

[5]Unless the noise is so large that the output state no longer represents the initial physical system: then all bets are off.

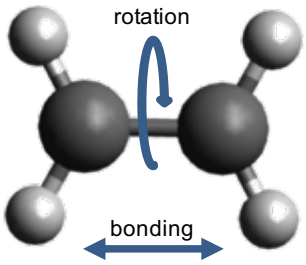

Figure 6: *Illustration of the ethylene rotation/torsion and bond stretching/breaking simulations.*

spins in all cases. In all cases, OpenFermion [21] is used to generate the circuits.

With a Unitary Coupled Cluster ansatz (see Section 4), the minimal representation needs to describe the rotation consisting of 4 qubits (representing 4 orbitals) and 2 terms in the wave function expansion that need to be optimized. Similarly, the bond breaking process requires 8 qubits and uses a wave function expansion with 14 parameters, the 4 sites Hubbard model requires 8 qubits and 9 parameters for a 2 electron occupancy; and 8 qubits with 14 parameters when simulating 4 electrons.

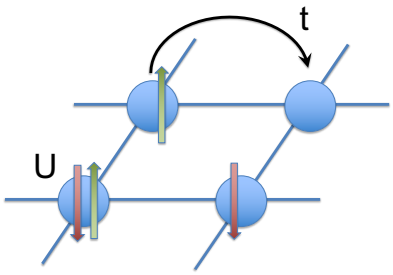

Figure 7: *Illustration of the 4-site Hubbard model simulation for electrons with spins, using hopping term t and Coulomb term U.*

6.1 Experimental Setup

Noise Injection: We extended the ProjectQ [30] quantum simulation infrastructure with noise injection capabilities. For each gate in the circuit circuit ($R_X(\theta)$, $R_Y(\theta)$, H, $CNOT$[6]), we add an operator in the form of rotations whose angles are independently sampled from a distribution: systematic over/under rotation (along the same axis) and noise drawn from a Gaussian probability distribution (main component along same axis, small orthogonal component). The noise operator for each gate is sampled independently of the others. For each scenario we perform sweeps with increasing noise strength until it breaks the minimizers. In the rest of this paper, numerical values for noise magnitude refer to the standard deviation (σ) of the Gaussian noise probability distribution.

Methodology: In each study, the minimizer is given an appropriate budget (maximum number of invocations of the objective function) and convergence criteria are adjusted in favor of using

[6]We do not add noise to $R_Z(\theta)$ as these are purely mathematical, thus noise-free.

up the budget. The minimizers are run until any convergence criteria are met or the budget is used up. We repeat the full algorithm several times and report the average and overall minimum across all runs, as well as the average result when running the simulation at the optimal parameters found. The results are compared to the results of classical ab-initio calculations.

Optimizer Baseline: The optimizers included in SCIKIT-QUANT have been described in Section 3. Each optimizer has been individually tuned with good settings for the type of problems generated by our VQE test circuits, see Section 3.5. As baseline comparison, we choose BFGS and Cobyla, both from SciPy [29], because they are well known and widely used, as explained in Section 2.2.

Hardware: The simulations were small enough, memory-wise, to run on a standard server. We note that for this study simulating the quantum circuit constitutes the main bottleneck; optimizers can run well and handle a large number of parameters when using just a single server.

6.2 Optimization Solution Quality

One of the effects of stochastic noise is to lift the results returned from the objective function as explained in Section 5 and shown in Figures 1 and 5. There are two ways to evaluate the optimizers: 1) by the minimum energy they actually find relative to what was possible given the response limitations of the objective function; or 2) by the quality of the optimal parameters found, evaluated by calculating the expected energy from a noise-free simulation run at those parameters. Which quality measure is most relevant will depend on the application and science goals at hand, so we provide examples of both. For example, in the case of chemistry studies, quantum subspace expansion [19] requires accurate parameters.

Distance to minimum energy. Figure 8 shows the average calculated energy of the full VQE algorithm for the ethylene rotation (left) and bond breaking simulation (right), for 100 runs at each noise level for the former and 10 each for the latter.[7] The straight, dashed, black lines show the chemical accuracy (0.00159 hartrees): a solution closer to the exact value than this cut-off (i.e. results below this line) are scientifically useful. The dashed yellow lines show the lowest value the objective function returned across all runs, i.e. the lowest value any of the minimizers could theoretically have found. Where this line is above the chemical accuracy, *the optimizer is not the weak link of the algorithm, the quantum processor is the limiting component.* The larger, deeper, 8-qubit circuit clearly suffers more from noise: even at moderate levels, a chip with such gate noise would be the weak link in the full algorithm.

Considering the minimizers, BFGS can not find the global minimum even with small levels of noise (lowest level shown is 10^{-4}), because it treats any gradients seen as real, including fakes due to noise, and gets stuck. It works, however, fine on a noise-free run (not plotted). The other baseline, Cobyla, performs quite well at low levels of noise, but clearly underperforms as noise increases. The optimizers designed to handle noise well outperform across the full range, with some stratification only happening at the highest noise levels and ImFil yielding the overall best results. In the low noise regime, however, where all optimizers perform similarly, other considerations, such as the total number of iterations, come into play to determine which is "best." Cobyla would then most likely be preferred (see Section 6.4 for a detailed discussion).

Parameter quality. Figure 9 (left) shows the results for the full VQE algorithm Hubbard model simulations, with the energy recalculated at the optimal parameters using a noise-free run. With the Hubbard model, the region of the optimization surface around the global minimum is rather shallow (see also Figure 2), which clearly stresses the optimizers a lot more. The behavior of BFGS and Cobyla mimics the results from the ethylene studies, but this time both NOMAD and especially SnobFit also underperform or even fail. A detailed analysis shows that this weakness is exposed by bounds that are too large for either optimizer to handle: reducing the bounds greatly improves their performance (whereas it does not for BFGS and Cobyla).

6.3 Leveraging Domain Science Constraints and Optimizer Knowledge

From the discussion above, it is already apparent that different methods perform best for different problems as optimization surfaces vary. Furthermore, the quality of the solution may be improved by exploiting a combination of domain science and optimizer knowledge. For our VQE examples, the most obvious and realistically actionable parameters are: 1) quality of initial solution; and 2) good parameter bounds.

Impact of initial solution quality. VQE for chemical problems has the advantage that a good initial can often be obtained from approximate classical calculations. To understand the impact of initial solution quality we consider a comparison of ImFil and PyBobyqa for the ethylene rotation simulation.

In Figure 11 we plot the evaluation points chosen by each optimizer: using a good initial at $(0.1, 0.1)$ and a bad one at $(0.3, -0.3)$. The global optimum is at $(0.00012, 0.04)$. Whether it receives a good (A) or bad (B) initial, ImFil will use the given bounds to determine its first stencil, doing a mostly global search. Although the initial drives the first few iterations, it quickly moves away from the bad initial, to converge at the optimum. PyBobyqa starts by considering only points within its trust region around the initial point. If the initial is close enough to make the global optimum fall within that region, it will find it quickly (C). However, if the initial is near a pronounced local minimum, in $(0.5, -0.5)$ in this case, it will get stuck (D), never finding the global minimum.

Overall, this analysis indicates that if good initials are available with low computational overhead, they can improve both the quality and speed to solution.

Impact of bounds. Some optimization methods, such as SnobFit, benefit greatly from having the search space (and thus the needed number of evaluations, alleviating scaling issues) reduced by tight bounds on the optimization variables. When possible, such bounds should be provided from the domain science. When bounds derived from first principles are unavailable, an automatic way of finding tighter bounds can be had by running a composition of optimizers. To illustrate this principle we show the effect of optimizer composition by using ImFil to derive tight bounds for SnobFit.

ImFil uses progressively smaller stencils in its search for the global minimum (see Section 3.2). Once close enough, the combination of high noise levels and a shallow optimization surface means

[7] The larger 8-qubit circuits took about two orders of magnitude more time to run.

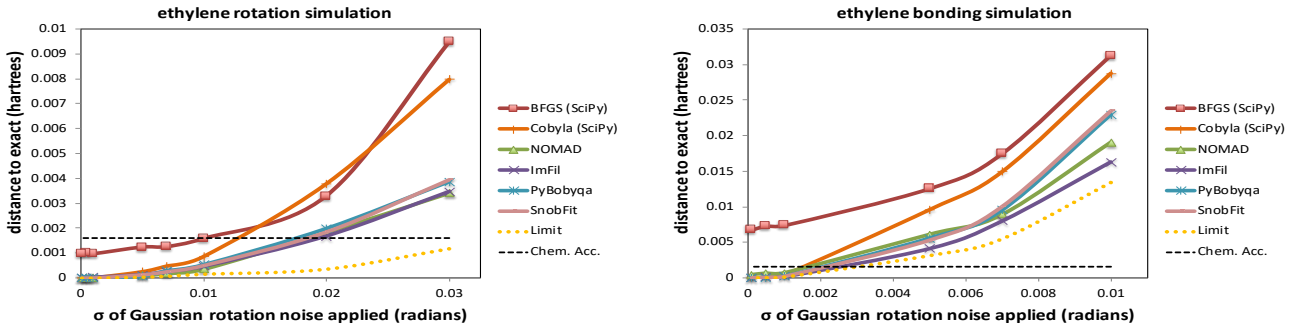

Figure 8: *Average calculated energy of the full VQE algorithm for the ethylene rotation (left) and bond breaking simulation (right). Lowest noise level is* 10^{-4}*. The cut-off for chemical accuracy is shown by the straight, dashed, black line. With increasing noise, the result from the objective function is increasingly moved away from the global minimum. The lowest value that the objective function could return at a given noise level is estimated by the dashed yellow line.*

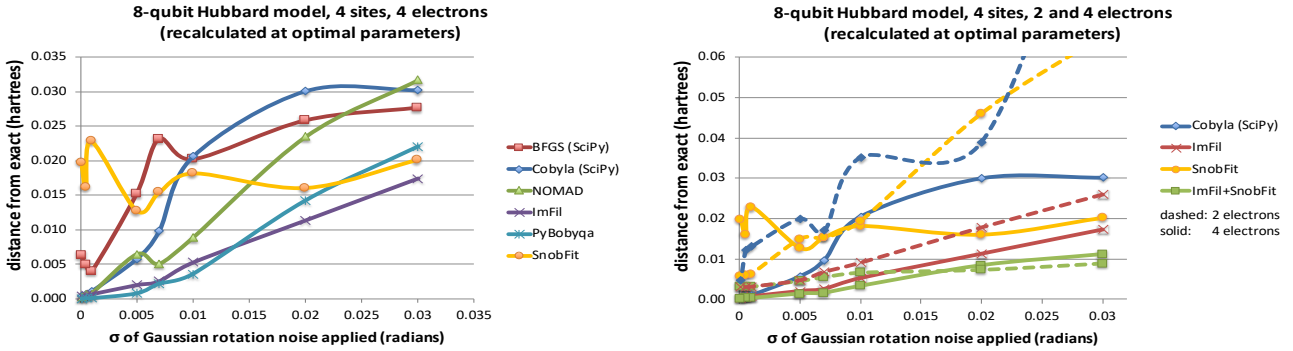

Figure 9: *Full VQE results of a 4 site Hubbard model simulation with 4 electrons for individual optimizers (left); and results when combining ImFil and SnobFit for 2 (dashed lines) and 4 (solid) electrons (right). The ground energy is recalculated at the optimal parameters found using a noise-free simulation.*

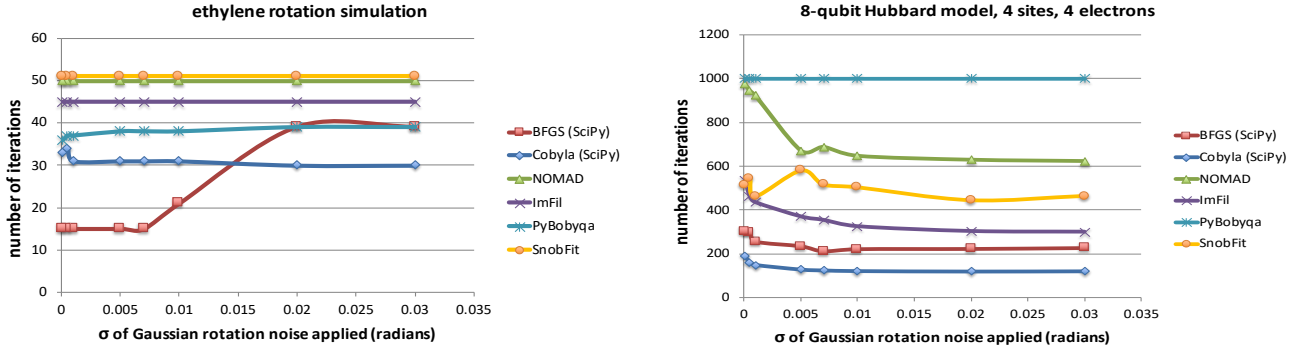

Figure 10: *Number of objective function calls used for each of the optimizers as a function of the noise level. Ethylene rotation simulation (left; budget of 50) and Hubbard Hubbard model simulation with 4 electrons (right; budget of 1000).*

that no further progress can be made on the stencil, which ImFil then labels as "failed." The last good stencil provides the necessary bounds for SnobFit to proceed and find a robust minimum. The results of this approach are shown in Figure 9 (right) for Hubbard simulations with occupancies of 2 and 4 electrons. In all cases, ImFil already outperforms the other optimizers, but SnobFit is still able to improve from the point where ImFil fails. Crucially, ImFil fails much earlier when noise levels are high (see Section 6.4), allowing the combined run of ImFil+SnobFit to stay within budget.

6.4 Performance Considerations

Besides finding a good solution, optimizer quality is also quantified by its total execution time. First, we note that for hybrid algorithms

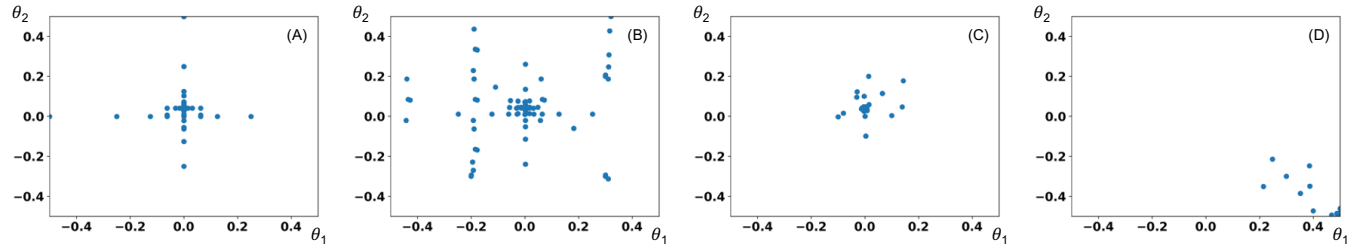

Figure 11: *Effect of the quality of the initial on the overall solution for the ethylene rotation simulation. Shown are parameters chosen by the optimizer to evaluate: ImFil with good (A) and with bad initial (B); PyBobyqa with good (C) and bad initial (D). The good initial is at* $(0.1, 0.1)$*, the bad at* $(0.3, -0.3)$*, and the global optimum is at* $(0.00012, 0.04)$.

the wall time is completely dominated[8] by the quantum chip for current devices. When considering the optimizer in isolation the number of objective function evaluations is thus a good proxy for wall clock performance.

Most optimizers provide control over the number of evaluations per iteration, thus determining single iteration overhead. We find in practice that the defaults work best: a certain minimum number of evaluations is always necessary to fill out a stencil, local model, or map a trust region. *The incremental improvement from adding more points to the current iteration is, however, less than the improvement obtained from spending that budget on an extra iteration.*

Convergence criteria provide control over the total number of iterations. Most optimizers define convergence as improvement between consecutive steps falling below a threshold, or failing altogether a given number of times. The lack of local improvement need not stop the search, e.g. for NOMAD and SnobFit it can be chosen to initiate more *global* searches, and subsequently use up the whole budget. Whether those global searches are useful depends on the quality of the initial and on the presence of local minima.

The setup of the science problem at hand matters greatly as well: tighter bounds and a higher quality initial reduce the number of iterations needed, as was already seen in Figure 11. An efficient ansatz with fewer parameters, for example through exploitation of symmetries, and an optimization surface with steep gradients near the global minimum, can also have a big impact.

Finally, there are differences intrinsic to the optimization methods. Figure 10 shows the number of objective function evaluations for increasing levels of noise, for both the ethylene rotation simulation (left) and the Hubbard model with 4 electrons (right). There is little sensitivity to noise in the much simpler rotation simulation, except for BFGS which falls apart at high noise levels. A clearer picture emerges in the Hubbard simulation: convergence criteria that take into account the observed level of noise in their definition of "no improvement" work best. E.g. PyBobyqa, which uses a fixed threshold, fails to converge, because noise causes sufficient differences between iterations to remain above threshold, so it continues, using up the full budget. The other optimizers, which either track overall improvement or improvement within an iteration given the noise, stop much earlier as noise increases. This is especially beneficial when conserving budget is important to allow switching of optimizers, e.g. from ImFil to SnobFit as shown in the previous section, while remaining within the budget overall.

[8]The true ratio depends on the quantum hardware chosen and the server CPU running the classical optimizer. We estimate the time spent in the classical step to be about 1% of the total. Furthermore, several of the optimizers are in pure Python and their wall clock performance could be greatly improved with a rewrite in C++ if necessary.

7 DISCUSSION

Much work is being dedicated to improving the VQE quantum circuits (depth, CNOT count, ansatz etc.) and to demonstrate science results on NISQ hardware. The need for noise-aware minimizers has been previously acknowledged, but its magnitude may have been understated. In fact, our study indicates that using a classical optimizer that is not noise-aware would make it the weakest link in the VQE chain: *use of specialized noise robust optimizers is essential on NISQ hardware.*

Our evaluations of the noise-aware optimizers we collected (and rewrote in some cases) into SCIKIT-QUANT indicate that:

- When solving noise-free optimization problems, SciPy optimizers such as BFGS or Cobyla are fastest by far. They do fail in the presence of even small noise, to the point of becoming unusable.
- When decent parameter bounds are available, ImFil is preferable, followed by NOMAD. When tight bounds are available, SnobFit should be considered. A composition of optimizers works best for final solution quality, e.g. running ImFil first to derive tight bounds for SnobFit.
- When high quality initial parameters are available, trust region methods such as PyBobyqa are fastest and preferable, followed by NOMAD and to a lesser extent SnobFit. ImFil is not sensitive to the value of the initial solution.
- Taking performance data into account does not change the above recommendations. We do note that some optimizers are adaptive and properly reduce the number of evaluations in the presence of noise, e.g. ImFil and NOMAD.
- When examining control over the number of iterations and search strategy (balancing solution quality, execution time, and premature convergence), ImFil provides direct control over scales and searches. For the others, only limited control is possible by tweaking the convergence criteria, (attenuated) step sizes, points in the local model, or overall budget.

Given our collection of optimizers, we wanted to know which method best handles the combination of optimization surfaces generated by the science problems and noise caused by the quantum hardware. Since the ansatz in VQE directly drives the former, and

influences the latter (e.g. through circuit depth), this provides important feedback for practical ansatz design. There are strong convergence requirements on the minimizer in terms of distance to the global minimum [20], but also constraints on the number of evaluations possible before convergence as e.g. calibrations may drift over the duration of the experiment. To make progress, the optimizer may need to find gradients on a surface with many local minima due to the noise, and do so with the least number of iterations possible. Our results support the following conjectures:

- There is no free lunch: a *suite* of minimizers is needed to match specific strengths to specific problems, making use of available domain science information such as a high quality initial parameters, knowledge of local minima, or the need to search around inaccessible regions.
- Circuit level noise redistributes counts in the output bit string probability distributions, from which the objective is calculated. This redistribution affects the latter in a non-linear way and thus does not simply average out. With large noise, it may thus be impossible to retrieve the actual global minimum value, but by searching for a robust minimum, the correct optimal parameters may still be found.
- For complex surfaces with local minima close to the global minima, noise can prevent the optimizer from distinguishing local from global. An understanding of the science is then needed to provide more constraints, e.g. in subdividing the problem and studying the minimum found in each with higher statistics.
- Most of the methods can scale up to hundreds of parameters. On NISQ hardware, with the minimizers provided, we expect the performance of hybrid approaches to be limited by the quantum part of the algorithms. The optimizers can easily execute on a single node server systems, no distributed memory parallelization is required yet.

Overall, this study indicates that the success of VQE on NISQ devices is contingent on the availability of classical optimizers that handle noisy outputs well at the scale of the "necessary" qubit concurrency. As of yet, this is a largely open research area, where our study details some of the challenges to be expected. Our software optimizers toolkit is directly useful to VQE Quantum Information Science practitioners, as well as a good starting point for mathematicians in search of better optimization methods tailored to VQE and other hybrid quantum-classical algorithms.

8 RELATED WORK

Hybrid quantum-classical algorithms such as VQE and QAOA employ optimizers in the classical part of the computation. For VQE, an initial discussion about optimization challenges in the presence of noise is provided by McClean et al. [20]. They study a unitary coupled cluster wavefunction for H_2, encoded into 4 qubits and with optimization over a single parameter. In the experiments, simulated measurement estimator noise is added to the objective function at a specified variance ϵ^2. They compare Nelder-Mead with TOMLAB/GLCLUSTER, TOMLAB/LGO, and TOMLAB/MULTIMIN. The choice of TOMLAB is motivated by the optimization study by Rios et al. [27], which reports a good combination of scalability and quality of solution. Even for this single parameter problem, these optimizers face challenges in the presence of noisy output. Current QAOA [35] studies still use BFGS and Nelder-Mead, as they still concentrate mostly on the quantum algorithm part of the problem. While the VQE result (system energy) is subject to physical or chemical laws which constrain its values, there is no such equivalent for most QAOA approaches. Thus, it is our expectation they will need to be supplemented with optimizers robust in the presence of noise.

An orthogonal approach in the realm of hybrid-algorithm design for short-depth circuits is the incorporation of error mitigation techniques. The proposed zero-noise extrapolation techniques [18, 31] seem to impose no constraints on optimizers and just run in the first step the full VQE algorithm. An additional step calibrates the impact of system noise, followed by an offline procedure to extrapolate results to the ideal regimen of zero-noise. While the IBM studies [18, 31] insert noise at the pulse level, Dumitrescu et al. [8] insert noise using additional CNOT gates and describe a zero-noise extrapolation procedure. Current results are for small circuits with few parameters (two) involved in the optimization. Their applicability to higher dimensional problems on complex optimization surfaces remains to be seen and whether they relax the requirements on robust optimizers.

Another area of interest is the work in the numerical optimization realm. Rios et al. [27] provide a comprehensive evaluation of derivative-free numerical optimizers along multiple dimensions including scalability and quality of solution, for convex and non-convex, smooth and non-smooth surfaces. Overall, they recommend the commercial TOMLAB [32] implementations of GLCLUSTER, LGO and MULTIMIN. Each is best for a given combination of surface convexity and smoothness. Also note that all the algorithms included in SCIKIT-QUANT are very close to any of the TOMLAB implementations for some type of surface.

9 CONCLUSION

Successful application of hybrid-quantum classical algorithms, with the classical step involving an optimizer, on current hardware, requires the classical optimizer to be noise-aware. We have collected a suite of optimizers in SCIKIT-QUANT that we have found to work particularly well, easily outperforming optimizers available through the widely used standard SciPy software.

We have focused on VQE, but we expect the results to be generally applicable: by providing a suite of optimizers with consistent programming interfaces, it is possible to easily apply combinations of optimizers, playing into their respective strengths. Our studies indicate that with these optimizers, the classical step is no longer the weakest link on NISQ-era hardware.

ACKNOWLEDGMENT

This work was supported by the DOE under contract DE-5AC02-05CH11231, through the Office of Advanced Scientific Computing Research (ASCR) Quantum Algorithms Team and Accelerated Research in Quantum Computing programs.

REFERENCES

[1] Abraham, H. *et. al.* Qiskit: An open-source framework for quantum computing, 2019.

[2] M. Abramson, C. Audet, J. Dennis, Jr., and S. Le Digabel. Orthomads: a deterministic mads instance with orthogonal directions. *SIAM Journal on Optimization*, 20(2):948–966, 2009.

[3] C. Audet and J. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.

[4] C. Audet and J. Dennis, Jr. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(1):445–472, 2009.

[5] C. Cartis, J. Fiala, B. Marteau, and L. Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. Technical report, University of Oxford, 2018.

[6] C. Cartis, L. Roberts, and O. Sheridan-Methven. Escaping local minima with derivative-free methods: a numerical investigation. Technical report, University of Oxford, 2018.

[7] A. Costa and G. Nannicini. Rbfopt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, Dec 2018.

[8] E. Dumitrescu, A. McCaskey, G. Hagen, G. Jansen, T. Morris, T. Papenbrock, R. Pooser, D. Dean, and P. Lougovski. Cloud quantum computing of an atomic nucleus. *Physical Review Letters*, 120(21), May 2018.

[9] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm, 2014.

[10] R. P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21:467–488, June 1982.

[11] F. Gao and L. Han. Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, Jan 2012.

[12] Google. Cirq. Available at https://github.com/quantumlib/Cirq.

[13] N. Hansen. Cma-es, covariance matrix adaptation evolution strategy for non-linear numerical optimization in python. Available at https://pypi.org/project/cma/.

[14] N. Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016.

[15] W. Huyer and A. Neumaier. Snobfit - Stable Noisy Optimization by Branch and Fit. *ACM Trans. Math. Software*, 35(9), 2008.

[16] C. Kelley. *Implicit Filtering*. SIAM, 2011.

[17] S. Le Digabel. Nomad: Nonlinear optimization with the mads algorithm. *ACM Trans. Math. Softw.*, 37(4):44:1–44:15, Feb. 2011.

[18] Y. Li and S. C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X*, 7(2), Jun 2017.

[19] J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven. Decoding quantum errors with subspace expansions. *Nature Communications*, 11(1), Jan 2020.

[20] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):23023, 2016.

[21] McClean, J. *et. al.* Openfermion: The electronic structure package for quantum computers, 2017.

[22] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[23] M. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Report DAMTP 2009/NA06., Department of Applied Mathematics and Theoretical Physics, Cambridge University, 2009.

[24] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994.

[25] W. Powell. Clearing the jungle of stochastic optimization. *INFORMS Tutorials in Operations Research: Bridging Data and Decisions*, Published online: 27 Oct 2014:109–137, 2014.

[26] R. Regis and C. Shoemaker. Combining radial basis function surrogates and dynamiccoordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[27] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, Jul 2013.

[28] Scikit-quant. scikit-quant.org.

[29] Scipy documentation. https://www.scipy.org.

[30] S. Steiger, T. Häner, and M. Troyer. ProjectQ: An Open Source Software Framework for Quantum Computing. *ArXiv e-prints*, Dec. 2016.

[31] K. Temme, S. Bravyi, and J. M. Gambetta. Error mitigation for short-depth quantum circuits. *Physical Review Letters*, 119(18), Nov 2017.

[32] Tomlab Optimization. TOMLAB. Available at https://www.tomopt.com/tomlab.

[33] J. J. Wallman and J. Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Phys. Rev. A*, 94:052325, Nov 2016.

[34] Z. Wang, S. Hadfield, Z. Jiang, and E. G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2), Feb 2018.

[35] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, 2018.