Reducing measurement costs by recycling the Hessian in adaptive variational quantum algorithms

IOP Publishing Quantum Sci. Technol. 10 (2025) 015031 https://doi.org/10.1088/2058-9565/ad904e

Quantum Science and Technology

OPEN ACCESS

RECEIVED
23 July 2024

REVISED
18 October 2024

ACCEPTED FOR PUBLICATION
8 November 2024

PUBLISHED
18 November 2024

PAPER

Reducing measurement costs by recycling the Hessian in adaptive variational quantum algorithms

Mafalda Ramôa[1,2,3,4,5,*], **Luis Paulo Santos**[3,4,5], **Nicholas J Mayhall**[2,6], **Edwin Barnes**[1,2] and **Sophia E Economou**[1,2]

[1] Department of Physics, Virginia Tech, Blacksburg, VA 24061, United States of America
[2] Virginia Tech Center for Quantum Information Science and Engineering, Blacksburg, VA 24061, United States of America
[3] International Iberian Nanotechnology Laboratory (INL), Braga, Portugal
[4] High-Assurance Software Laboratory (HASLab), Braga, Portugal
[5] Department of Computer Science, University of Minho, Braga, Portugal
[6] Department of Chemistry, Virginia Tech, Blacksburg, VA 24061, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: mafalda@vt.edu

Keywords: quantum computing, quantum chemistry, ADAPT-VQE, variational quantum eigensolver, NISQ algorithms

Abstract

Adaptive protocols enable the construction of more efficient state preparation circuits in variational quantum algorithms (VQAs) by utilizing data obtained from the quantum processor during the execution of the algorithm. This idea originated with Adaptive Derivative-Assembled Problem-Tailored variational quantum eigensolver (ADAPT-VQE), an algorithm that iteratively grows the state preparation circuit operator by operator, with each new operator accompanied by a new variational parameter, and where all parameters acquired thus far are optimized in each iteration. In ADAPT-VQE and other adaptive VQAs that followed it, it has been shown that initializing parameters to their optimal values from the previous iteration speeds up convergence and avoids shallow local traps in the parameter landscape. However, no other data from the optimization performed at one iteration is carried over to the next. In this work, we propose an improved quasi-Newton optimization protocol specifically tailored to adaptive VQAs. The distinctive feature in our proposal is that approximate second derivatives of the cost function are recycled across iterations in addition to optimal parameter values. We implement a quasi-Newton optimizer where an approximation to the inverse Hessian matrix is continuously built and grown across the iterations of an adaptive VQA. The resulting algorithm has the flavor of a continuous optimization where the dimension of the search space is augmented when the gradient norm falls below a given threshold. We show that this inter-optimization exchange of second-order information leads the approximate Hessian in the state of the optimizer to be consistently closer to the exact Hessian. As a result, our method achieves a superlinear convergence rate even in situations where the typical implementation of a quasi-Newton optimizer converges only linearly. Our protocol decreases the measurement costs in implementing adaptive VQAs on quantum hardware as well as the runtime of their classical simulation.

1. Introduction

Despite the limitations of current quantum devices, there is still enormous interest in seeing whether they can provide quantum advantage in the simulation of strongly correlated quantum many-body systems. In particular, the variational quantum eigensolver (VQE) [1] was proposed as a near-term algorithm for many-body fermionic problems. In contrast with algorithms designed for the fault-tolerant quantum computing era, VQE employs shallow circuits and undertakes a naturally noise-resilient learning strategy, where a classical optimizer is used to tune parameters of quantum gates.

A crucial part of VQE is the ansatz: it impacts accuracy as well as near-term viability. Problem-agnostic options have been shown to induce trainability issues termed *barren plateaus* [2]. Additionally, the coherence time requirements of current ansätze are, for reasonable-sized problems, still beyond what today's quantum computers have to offer. As such, new approaches attempting to further decrease circuit depth and improve trainability have emerged.

One promising option is to build the ansatz adaptively, such that its structure is dictated by the problem at hand. The first algorithm to use such a strategy was the Adaptive Derivative-Assembled Problem-Tailored VQE (ADAPT-VQE), proposed in [3] and extended in [4–9]. Starting from a very simple reference state, ADAPT-VQE creates a circuit block-by-block using information available on the fly. In the simplest version, each iteration selects one anti-Hermitian operator from a pre-defined pool, based on the associated gradient magnitude. This operator is multiplied by a variational parameter and exponentiated to create a parameterized unitary operator, which is appended to the ansatz. A VQE subroutine is then employed to optimize all of the parameters. After a new addition, the initial parameter vector for the optimization is built by appending a zero to the final parameter vector from the previous optimization, such that the newly added unitary is initially the identity. ADAPT-VQE leads to shallow and problem-tailored ansätze, and its parameter value recycling strategy makes it resilient against local traps [10].

Inspired by ADAPT-VQE, other variational quantum algorithms (VQAs) have since employed this idea of iteratively growing the ansatz circuit as well as the gradient vector. An example is ADAPT-QAOA, proposed in [11] and extended/studied in [12–14]. Other algorithms employ a fixed ansatz structure, but still augment the parameter vector iteratively to alleviate local traps and help avoid barren plateaus [15, 16]. These strategies can be thought of as a special case of ADAPT-VQE where the selection criterion is trivial (pre-determined operators) but the parameter recycling strategy is employed regardless.

The choice of which classical optimizer to employ in a VQA is important, as it impacts the costs of the algorithm as well as the solution quality. BFGS, a quasi-Newton optimizer which uses approximate second derivatives of the cost function to inform the search direction while avoiding the (costly) requirement of explicitly measuring them [17–20], is a popular choice [3, 5, 21–23]. A typical implementation of this algorithm will initialize an approximate inverse Hessian H, a matrix with all second derivatives, at the identity. Curvature information is gathered as the iterations proceed by performing rank-2 updates on this matrix, which are based solely on changes in the gradient and parameter vectors. As H is updated, the search direction evolves from a vanilla gradient descent into an approximation to Newton's direction.

In a typical implementation of an adaptive VQA with the BFGS optimizer, the approximate inverse Hessian is reinitialized to the identity matrix each time the ansatz is extended. The reinitialization of H seems discordant with the recycling of the parameters: The initial state at iteration n is exactly the same as the final state from iteration $n-1$, and thus all previous approximate second derivatives remain as accurate as before.

In this work, we propose a strategy to recycle the Hessian in adaptive VQAs. At the end of each VQE subroutine, we save the collected curvature information and use it to inform the initialization of H for the next optimization. With our protocol, the state of the optimizer, with all its knowledge of the cost landscape, is transferred between optimizations instead of being erased and restarted from scratch. We show that as a result, the search direction aligns more quickly with Newton's direction, and superlinear convergence is attained more often. Surprisingly, recycling old second derivatives allows the optimizer to better approximate new ones too, allowing new correlations to be captured earlier in the optimization. Our strategy leads to a significant decrease in the number of function evaluations required per optimization, which translates to a decrease in the number of calls to the quantum processor. For the molecules we studied, with 12 and 14 qubits, our protocol reduces the total measurement cost of the algorithms by an order of magnitude. We expect the impact to become more significant for larger system sizes.

The measurement costs are one of the most critical limitations of adaptive VQAs. They may be alleviated in two ways: by decreasing the number of measurements required to obtain the pool gradients, or by decreasing the number of measurements required in the optimization. Previous works have focused on the former [7, 24], such that the cost bottleneck now resides in the optimization process. To our knowledge, the strategy we propose is the first to tackle this source of costs, and the first to propose an optimizer specifically tailored to adaptive VQAs. We note that its relevance is not limited to implementations on quantum hardware—it also reduces the runtime of classical simulations of these algorithms, which can take hours even for relatively small molecules such as H_6 on a minimal basis set (12 qubits).

This paper is organized as follows. In section 2, we cover the relevant background topics: ADAPT-VQE and gradient-based optimization methods. In section 3 we introduce our main proposal, a BFGS algorithm tailored to ADAPT-VQE. In section 4, we present results from numerical simulations proving that our

method is capable of significantly reducing the total number of calls to the quantum computer required by ADAPT-VQE. We additionally examine quantitative markers along specific optimizations to justify the remarkable speed up in convergence, and seek to understand under which conditions the advantage is the greatest by analyzing the distance between the initial approximate inverse Hessian and the initial exact inverse Hessian, as well as the difference between the latter and the final exact inverse Hessian. We conclude in section 5.

2. Background

2.1. Variational quantum algorithms

VQAs are hybrid quantum–classical algorithms whose goal is to minimize a cost function. This cost function is typically formulated as the expectation value of a quantum Hamiltonian $\hat{\mathcal{H}}$, which can be written as a linear combination of Pauli strings and measured in a quantum computer via sampling [25].

The role of the quantum computer is then to host trial states that can be measured to evaluate the cost function. Such states are prepared by a parameterized quantum circuit, the ansatz, which is usually comprised of two parts: the preparation of a reference state $|\psi_{ref}\rangle$ and the application of a parameterized unitary $U(x)$. x is a vector whose elements map to gate parameters (e.g. angles of single-qubit rotation gates). In the interest of clarity, we will always denote parameter vectors by x and the individual entries by θ_i, such that if our parameter vector is n-dimensional we have $x = \{\theta_1, \ldots, \theta_n\}$. This allows us to clearly distinguish iterations of the numerical optimizer, where x_k is used to denote the parameter vector at iteration k, from iterations of adaptive VQAs, where θ_i is used to denote the parameter added at iteration i.

For a given ansatz and cost function, a VQA will seek the state corresponding to the lowest value of the cost function within the search space defined by the ansatz. This minimization is done by a classical optimizer.

The VQE is a subclass of VQAs aimed at finding eigenstates and eigenvalues of physical systems [1]. We focus on quantum chemistry and discuss VQEs for finding molecular ground states under the Born-Oppenheimer approximation.

The first ansatz proposed for this problem was the Unitary Coupled Cluster Singles and Doubles (UCCSD) ansatz, motivated by classical variational methods for quantum chemistry [26, 27]. In recent years, strategies for growing the ansatz adaptively (an idea first proposed in ADAPT-VQE [3]) have gained popularity. They have been shown to lead to shallower circuits, higher accuracy, and improved resilience against local traps [28]. Our work is aimed at such adaptive algorithms. A detailed description of the workflow of ADAPT-VQE in provided in section 2.2.

In order to solve the electronic structure problem using a quantum computer, we need a fermion-to-qubit mapping. A popular choice is the Jordan-Wigner transform [29], given by

$$\begin{aligned} a_i^\dagger &\to \frac{1}{2}\prod_{k=1}^{i-1} Z_k \cdot (X_i - iY_i)\,, \\ a_i &\to \frac{1}{2}\prod_{k=1}^{i-1} Z_k \cdot (X_i + iY_i)\,, \end{aligned} \tag{1}$$

where Z_k, X_i, Y_i are Pauli operators acting on the qubits labeled by the respective indices. $a_i^\dagger$ (a_i) is the creation (annihilation) operator for orbital i. This transformation can be used to map fermionic Hamiltonians to quantum-mechanical observables, as well as to transform fermionic state preparation unitaries to circuits.

2.2. ADAPT-VQE

We now introduce the relevant facets of the ADAPT-VQE algorithm, proposed in [3].

2.2.1. Algorithm

The idea behind ADAPT-VQE is to let the molecule under study 'choose' its own state preparation circuit, by creating the ansatz in a strongly system-adapted manner. Pseudo-code for this protocol can be found in algorithm 1. We use * to denote optimized values.

Algorithm 1. ADAPT-VQE.

Input:
$|\psi^{(ref)}\rangle$, $\{\hat{A}_k\}_K$, $\hat{\mathcal{H}}$ ▷ Problem specification
ϵ, L ▷ Hyperparameters
Output:
$|\psi^*\rangle$, x^*, E^*
1 $n \leftarrow 0$;
2 $x_0 \leftarrow \{\}$;
3 $|\psi^{(n)}\rangle \leftarrow |\psi^{(ref)}\rangle$;
4 $E_n \leftarrow$ measure_energy $\left(\hat{\mathcal{H}}, |\psi^{(n)}\rangle\right)$;
5 **while** $n < L$ **do**
6 $n \leftarrow n + 1$;
7 **for** $k \leftarrow 1 \ldots sK$ **do**
8 Measure $g_k = \langle\psi^{(n-1)}| \left[\hat{\mathcal{H}}, \hat{A}_k\right] |\psi^{(n-1)}\rangle$; ▷ Measure pool gradients
9 $i \leftarrow k$ s.t. $g_k = \max(\{|g_k|\}_K)$ ▷ Select new operator
10 $G \leftarrow \|\{g_1, \ldots, g_K\}\|_F$; ▷ Calculate gradient norm
11 **if** $G > \epsilon$ **then**
12 $|\psi^{(n)}\rangle \leftarrow e^{\theta_i \hat{A}_i} |\psi^{(n-1)}\rangle$; ▷ Grow ansatz
13 $x_n \leftarrow \{x_{n-1}, 0\}$; ▷ Grow parameter vector
14 $E_n, x_n \leftarrow \text{VQE}(\hat{\mathcal{H}}, |\psi^{(n)}\rangle, x_n)$; ▷ Minimize energy
15 **else**
16 Return $|\psi^{(n-1)}\rangle$, x_{n-1}, E_{n-1}; ▷ Successful convergence
17 Return $|\psi^{(n)}(x)\rangle$, x_n, E_n; ▷ Convergence target unmet

Here, ϵ, L are user-specified hyperparameters that define termination. The algorithm stops when the norm of the pool gradients is below ϵ or when a maximum number of iterations L is reached, whichever happens first.

The role of the user in the creation of the ansatz is the selection of a size K *operator pool* $\{\hat{A}_k\}_K$. This pool contains the generators of the unitary operators that may be added to the ansatz. Circuits for these unitaries may be created using ladders-of-CNOTs [30] (along with Trotterization if the corresponding Pauli strings do not commute), or pool-specific protocols [31].

The ansatz is initialized to identity: In the first iteration, the prepared state is simply the (classical) Hartree–Fock ground state. Each iteration adds an operator to the ansatz, along with the corresponding variational parameter, initialized at zero. Thus, the state preparation circuit and the parameter vector grow from iteration to iteration.

The selected pool operator is the one which leads to the derivative $\frac{\partial E^{(n)}}{\partial \theta_k}$ of greatest magnitude at point $\theta_k = 0$. These derivatives can be written as an expectation value using the formula

$$\partial E^{(n)} \theta_k \Big|_{\theta_k = 0} = \left\langle \psi^{(n-1)} \middle| \left[\hat{\mathcal{H}}, \hat{A}_k\right] \middle| \psi^{(n-1)} \right\rangle. \tag{2}$$

In each iteration n, the VQE subroutine (step 14) minimizes the energy with respect to a fixed structure ansatz $|\psi^{(n)}\rangle$ containing n variational parameters. The initial point for the optimization is the previous iteration's optimized vector augmented with a zero (step 14), which has been shown to improve trainability and resilience against local minima as compared to random initialization [28].

This parameter value recycling strategy can be used independently of the rest of the algorithm. We can bypass the dynamic circuit creation and grow it in a predefined manner, albeit still augmenting the parameter vector iteratively. This approach falls within the realm of optimization strategies rather than ansatz design, but it can be seen as an adaptive state preparation scheme with a trivial selection criterion. This has been successfully applied to tasks such as classifying hand-written digits with quantum neural networks [15] and finding the maximum cut value on a graph using the quantum approximate optimization algorithm (QAOA) [16].

2.2.2. Operator pool

The operator pool restricts the type of ansätze ADAPT-VQE can construct, and thus is the most important user-defined aspect of the algorithm. Currently, the most hardware-efficient operator pools are the qubit excitation (QE) pool [5] and the qubit pool [4].

The qubit excitation pool is comprised of two- or four-qubit operators which preserve particle number and Z spin projection (S_z), but do not respect the fermionic anticommutation relations. An example QE acting on four spin-orbitals p, q, r, s is

$$\begin{aligned}\hat{\tau} = i\big(&-X_qX_pX_sY_r - X_qX_pY_sX_r\\ &+X_qY_pX_sX_r - X_qY_pY_sY_r\\ &+Y_qX_pX_sX_r - Y_qX_pY_sY_r\\ &+Y_qY_pX_sY_r + Y_qY_pY_sX_r\big).\end{aligned} \tag{3}$$

Efficient circuit implementations for QE evolutions were proposed in [31].

Qubit pools [4] are pools in which each operator consists of an individual Pauli string. They do not conserve particle number or S_z in general, nor do they respect anticommutation. The corresponding evolutions are straightforwardly implemented using ladder-of-CNOTs circuits [30]. We consider the qubit pool formed from all individual Pauli strings appearing in the QE pool.

These two pools define two subclasses of the ADAPT-VQE algorithm: the Qubit Excitation Based (QEB)-ADAPT-VQE [5] and the Qubit-ADAPT-VQE [4]. We note that the former was proposed with a few possible algorithmic modifications in addition to the choice of pool. However, since such modifications are outside of the scope of this work, we take it to be the canonical ADAPT-VQE protocol (as defined in algorithm 1) implemented with the QE pool.

2.2.3. Measurement costs

One of the main limitations of adaptive VQAs is the scaling of the measurement costs. These costs come from two components: the VQE subroutine (step 1) and the measurement of the pool gradients (step 14). A brief discussion of these costs follows.

The molecular Hamiltonian of a system represented by N spin-orbitals (qubits) will have $\mathcal{O}(N^4)$ terms, so this is the worst-case measurement cost of an energy evaluation. However, empirical evidence shows that the Pauli strings in molecular Hamiltonians may be grouped into commuting sets of linear size, resulting in an $\mathcal{O}(N^3)$ cost for each energy evaluation (which seems unlikely to be further decreased) [32, 33]. The total measurement cost of one optimization is the cost of one energy evaluation multiplied by the total number of energy evaluations, which might come directly from energy measurements or indirectly from gradient measurements (measuring a length n gradient vector comes at a cost of $2n$ energy evaluations; see appendix A for a review on how to measure gradients on hardware).

As for the cost of measuring the gradients (step 14), it is $\mathcal{O}(N^5)$ for both the qubit and QE pools [24]. This means that the bound on the optimization costs is higher than the bound on the gradient measurement costs if the number of energy measurements per optimization grows faster than quadratically with N. Typical implementations of line searches require evaluating the gradient vector at least once (see appendix B.1 for a description of a line search algorithm). In this case, the optimization costs will dominate if the number of optimizer iterations (line searches) grows faster than linearly with N. We verify this is the case in numerical simulations.

As such, we believe that strategies to expedite the optimization—such as ours—tackle the biggest source of measurement costs in this algorithm as of now. We confirm this numerically in the results section (table 1).

2.3. Gradient-based optimization methods

The choice of classical optimizer is pivotal in a VQA: it impacts not only the quality of the output solution, but also the costs, since different optimizers will require different numbers of calls to the quantum computer. We refer to [34, 35] for an overview of the topic.

We focus on numerical optimization methods which explicitly use the gradient vector when setting the search direction. We refer to appendix A for a discussion on how to evaluate the gradient when the cost function is evaluated using a quantum computer.

Gradient-based optimization methods employ a succession of line searches, using the final point of each as the initial point for the next. p_k, the search direction at iteration k, is determined from data collected at x_k, the initial point. Such data includes, but is not necessarily limited to, the gradient vector. The optimizer then seeks a step size α which (perhaps approximately) minimizes the cost function f along p_k. The next iterate is set as

$$x_{k+1} = x_k + \alpha p_k. \tag{4}$$

In the next sections, we discuss different strategies to choose p_k.

2.3.1. Gradient descent

Vanilla gradient descent [20] is a first-order optimization method where the search direction at iteration k is opposite to the gradient at x_k, i.e.

$$p_k^{(\mathrm{GD})} = -\nabla f(x_k)\,. \tag{5}$$

This means that steps are taken in the direction of steepest descent. Note that there always exists an $\alpha > 0$ such that this step direction produces a lower value of f at the next iterate (equation (4)). Since the gradient is not suggestive of a particular value for α, heuristics must be used.

Gradient descent usually takes significantly longer to converge than more sophisticated alternatives.

2.3.2. Newton's method

Newton's method [19, 20, 36] employs a second-order modification to the gradient descent direction using $\nabla^2 f$, the Hessian of f (a matrix containing all its second derivatives). Newton's direction,

$$p_k^{(N)} = -\nabla^2 f^{-1}(x_k)\,\nabla f(x_k)\,, \tag{6}$$

is the vector pointing at the minimum of a quadratic model of f at x_k, as given by a second-order Taylor expansion,

$$\begin{aligned} f(x_k + \Delta x) \approx f(x_k) + (\nabla f(x_k))^T \Delta x \\ + \frac{1}{2}(\Delta x)^T \nabla^2 f(x_k)\,\Delta x. \end{aligned} \tag{7}$$

If $\nabla^2 f(x_k)$ is positive definite, it admits an inverse and induces a convex quadratic model. Therefore, the positive definiteness of the Hessian guarantees that equation (7) has a minimum. If this condition does not hold, $p_k^{(N)}$ might be undefined or correspond to an ascent direction. This is more likely in regions farther away from a minimum, where a bowl-shaped approximation is bound to be less adequate. In such cases, a positive-definite modification of the Hessian will typically be used in its place.

Since a unit step size would lead to the minimum if f were quadratic in its variables, most practical implementations set α to one (with possible adjustments depending on the observed decrease of the cost function).

The use of second-order derivatives enables Newton's method to enjoy a quadratic convergence rate [37]. However, this comes at considerable cost: the evaluation of $\mathcal{O}(n^2)$ second derivatives and the inversion of a $n \times n$ matrix, where n is the optimization dimension.

2.3.3. Quasi-Newton methods

Quasi-Newton methods [17, 19, 20, 36] collect and use second-order information without explicitly evaluating the second derivatives of f. In general, they converge slower than Newton's method, but faster than gradient descent. Despite never computing the Hessian matrix explicitly, they often reach superlinear convergence rates. In these methods, the search direction is given by

$$p_k^{(\mathrm{QN})} = -H_k \nabla f(x_k)\,, \tag{8}$$

where H_k is an approximation to the inverse Hessian (the actual Hessian is usually denoted B_k in numerical optimization literature). This approximation is built and updated along the optimization, using available information that does not directly include second derivatives.

There are many quasi-Newton methods. We will focus on BFGS, since it is considered the most efficient [20] and has many desirable properties (see appendix B.3). This method was named after Broyden [38], Fletcher [39], Goldfarb [40] and Shanno [41] who proposed it independently in 1970. We provide pseudo-code for this optimizer in algorithm 2. While many variants exist, we choose to remain as close as possible to the implementation in Scipy's [42] numerical optimization submodule `optimize`, as it is widely used in practice.

f and its gradient ∇f must be supplied as callables, such that the optimizer can evaluate them for any parameter vector. Common choices for the initial point x_0 are all-zero, random, and problem-specific initializations.

The hyperparameters $M \in \mathbb{N}$, $\epsilon_o \in \mathbb{R}^+$ correspond respectively to the maximum number of iterations (line searches) and to a convergence threshold on the gradient norm. We use the subscript o to distinguish

Algorithm 2. BFGS.

Input:
$f, \nabla f, x_0$ ▷ Problem specification
M, ϵ_o ▷ Hyperparameters
Output:
minimizer $x^*, f(x^*)$
1 $n \leftarrow \text{length}(x_0)$
2 $H_0 \leftarrow I_{n\times n}$ ▷ Initialize inverse Hessian
3 $k \leftarrow 0$
4 **while** $k <$ M **do**
5 $p_k \leftarrow -H_k \nabla f(x_k)$; ▷ Update search direction
6 $x_{k+1}, f(x_{k+1}), \nabla f(x_{k+1}) \leftarrow$ `line_search`$(p_k, f, \nabla f, x_k, f(x_k), \nabla f(x_k))$;
7 **if** $\|\nabla f(x_{k+1})\|_F > \epsilon_o$ **then**
8 $s_k \leftarrow x_{k+1} - x_k$;
9 $y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$;
10 $H_{k+1} \leftarrow$ `update_h`(H_k, s_k, y_k); ▷ Update inverse Hessian
11 $k \leftarrow k + 1$;
12 **else**
13 Return $x_{k+1}, f(x_{k+1})$; ▷ Successful convergence
14 **end**
15 **end**
16 Return $x_k, f(x_k)$; ▷ Convergence target unmet

this threshold from the ADAPT-VQE convergence threshold in algorithm 1. The optimization is stopped when the number of iterations reaches M or when the magnitude of the gradient vector falls below ϵ_o, whichever happens first. Typical values for M, ϵ_o are on the order of 10^2 to 10^4, 10^{-6} to 10^{-8} respectively.

`line_search` is an algorithm which seeks the α that minimizes $f(x_{k+1})$, with x_{k+1} given by equation (4). We provide more details about this subroutine in appendix B.1.

In the first iteration of BFGS, the direction calculated in step 5 is opposite to the gradient (just like in gradient descent algorithms). This is due to the choice of setting H_0 to the identity. In principle, this could be any symmetric positive definite matrix; however, in general, heuristics for how to choose it are lacking. The identity matrix is the standard, unbiased option, and it is usually assumed by numerical optimizers (this is the case in SciPy's implementation [42]).

As the algorithm proceeds, H_k will be updated to better reflect the curvature of f, thus refining the search direction. These updates depend only on parameter and gradient vectors. For more details we refer to appendix B.2.

2.3.4. Convergence rates

A brief discussion of convergence rates follows. We refer to [20] for details.

In this work, we will define convergence rates based on ratios of errors. This is referred to as 'Q-convergence' (from *quotient*) and is standard in numerical optimization literature.

The iterates $\{x_k\}$ are said to converge Q-linearly to the solution x^* if there is a constant $r \in (0, 1)$ such that

$$\frac{\|x_{k+1} - x^*\|_F}{\|x_k - x^*\|_F} \leqslant r \tag{9}$$

for all large enough k. They are said to converge Q-superlinearly if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|_F}{\|x_k - x^*\|_F} = 0\,, \tag{10}$$

and Q-quadratically if there is a constant $M \in \mathbb{R}^+$ such that

$$\frac{\|x_{k+1} - x^*\|_F}{\|x_k - x^*\|_F{}^2} \leqslant M \tag{11}$$

for all large enough k.

Q-linear convergence is typical of gradient descent, while Newton's method usually enjoys Q-quadratic convergence. Quasi-Newton methods are in between, often converging Q-superlinearly. More specifically, if the quasi-Newton Hessian B_k and search direction p_k obey

$$\lim_{k\to 0} \frac{\| \left(B_k - \nabla^2 f(x^*)\right) p_k \|_F}{\|p_k\|_F} = 0\,, \tag{12}$$

then there will be an index k_0 such that the unit step length will be accepted for all iterations $k > k_0$, and the iterates $\{x_k\}$ will converge to the solution Q-superlinearly (see theorem 3.6 in [20]).

The difference between the approximate and exact Hessians along the search direction going to zero as the iterations proceed (equation (12)) is both necessary and sufficient for quasi-Newton methods to be Q-superlinearly convergent.

As the Q-convergence definition will be used at all times, we will omit the 'Q-' when referring to convergence rates in what follows.

3. BFGS algorithm for ADAPT-VQE

Having discussed the BFGS optimizer as well as the ADAPT-VQE algorithm, we now propose an ADAPT-VQE-tailored BFGS optimizer in algorithm 3.

In labeling the inverse Hessian H, we use a single subscript k to denote the kth optimization iteration, and $n \times n$ to denote the nth ADAPT-VQE iteration (since this sets the dimension of the matrix). When either iteration label is clear from context, we omit the corresponding subscript to simplify the notation.

We note that aside from the callables f, ∇f, all inputs to the algorithm at iteration n are outputs from iteration $n-1$. The point $x^*_{(n-1)}$, as well as the corresponding gradient $\nabla f(x^*_{(n-1)})$ and inverse Hessian $H^*_{n-1\times n-1}$, pertain to the state of the optimizer at the end of iteration $n-1$.

The novelty of algorithm 3 is in step 3, where we use the inverse Hessian resulting from the $(n-1)$th ADAPT-VQE optimization to initialize the inverse Hessian at the nth optimization. We note that because the final inverse Hessian will be used as an input for the next iteration, we update it in line 10, before the convergence check. This is in contrast with algorithm 2, where the Hessian is not updated in the iteration where convergence is reached.

The motivation behind algorithm 3 is that ADAPT-VQE uses the final point of one optimization as the initial point for the next (see algorithm 1). Since at the start of the new optimization we have not moved in the cost landscape, the curvature information we gathered during the previous optimization continues to approximately capture the shape of the parameter landscape, at least in regards to the parameters $\theta_0, \ldots, \theta_{n-1}$. As we have no second-order information concerning θ_n yet, we choose to expand $H^*_{n-1\times n-1}$ with a unit diagonal and zeros elsewhere. This unbiased choice evidently preserves the positive definiteness of the matrix, thus our modified algorithm also guarantees that the quadratic model has a minimum and p_k corresponds to a descent direction at all times.

It is simple to see that positive definiteness is also preserved by the act of removing a row and the corresponding column from H_k. This means that we have the freedom to select from which parameters we wish to preserve second-order information. As a result, our strategy generalizes to the case where we wish to freeze a subset $\{\theta_{f_1}, \ldots, \theta_F\}$ of F parameters that were active in the previous iteration. In this case, we simply remove from $H^*_{n-1\times n-1}$ the rows and columns corresponding to the indices $f_1, \ldots, F$. This was not explicitly included in algorithm 3 for the sake of conciseness.

Equipped with our modified BFGS algorithm, ADAPT-VQE has the flavor of a continuous optimization of growing dimension. The search space is expanded when the ansatz gradient norm falls below ϵ_o, because we do not expect a significant energy decrease along the directions in the current cost landscape. The algorithm terminates when the pool gradient norm is below ϵ, because we do not expect a significant energy decrease along the directions in which we can expand this space.

The selection criterion of ADAPT-VQE dictates that the direction in which to expand the parameter space leads to the steepest cost landscape. This is reminiscent of first-order optimization methods, with the difference being that here we are screening possible candidates, and so the choice of direction is made with respect to parameters not currently in the parameter vector.

For the sake of conciseness, we will refer to our approach as *recycling the Hessian*. Note that the inverse Hessian is unique and fully determines the actual Hessian, such that by recycling one we implicitly recycle the other.

Algorithm 3. BFGS for ADAPT-VQE Iteration n.

Input:
$f, \nabla f, x^*_{(n-1)}, \nabla f(x^*_{(n-1)}), H^*_{n-1\times n-1}$ ▷ Problem specification
M, ϵ_o ▷ Hyperparameters
Output:
minimizer $x^*, f(x^*), \nabla f(x^*), H^*_{n\times n}$
$x_0 \leftarrow \{x^*_{(n-1)}, 0\}$
$g_0 \leftarrow \{\nabla f(x^*_{(n-1)}), \frac{\partial f}{\partial \theta_n}(x_0)\}$
$H_0 \leftarrow \begin{pmatrix} H^*_{n-1\times n-1} & 0 \\ 0 & 1 \end{pmatrix}$ ▷ Initialize inverse Hessian
$k \leftarrow 0$
while $k <$ M **do**
  $p_k \leftarrow -H_k \nabla f(x_k)$; ▷ Update search direction
  $x_{k+1}, f(x_{k+1}), \nabla f(x_{k+1}) \leftarrow$ `line_search`$(p_k, f, \nabla f, x_k, f(x_k), \nabla f(x_k))$;
  $s_k \leftarrow x_{k+1} - x_k$;
  $y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$;
  $H_{k+1} \leftarrow$ `update_h`(H_k, s_k, y_k); ▷ Update inverse Hessian
  $k \leftarrow k + 1$;
  if $\|\nabla f(x_k)\|_F < \epsilon_o$ **then**
    Return $x_k, f(x_k), \nabla f(x_k), H_k$; ▷ Successful convergence
  end
end
Return $x_k, f(x_k), \nabla f(x_k), H_k$; ▷ Convergence target unmet

4. Results

In this section, we present numerical simulation results for different systems at various bond lengths. While we focus on linear H_6 (12 qubits) as a toy model for a strongly correlated system, we additionally consider LiH (12 qubits) and BeH_2 (14 qubits) as real molecules. In all cases, we use the STO-3G basis set and no frozen-core approximations. The pools we use are those defined in section 2.2: the qubit excitation pool [5] and the qubit pool [4]. We set the ADAPT-VQE convergence threshold ϵ to 10^{-6} and 10^{-5}, respectively. The threshold for the qubit pool is set to a higher value because this pool is larger.

The code used for the numerical simulations has been made publicly available on GitHub (https://github.com/mafaldaramoa/ceo-adapt-vqe/tree/main). OpenFermion [43] was used for manipulating fermionic operators, and the corresponding plugin with PySCF [44] for the underlying electronic structure calculations. All expectation values were calculated via matrix algebra. To calculate distances between matrices, we use the Frobenius norm, defined for an $m \times n$ matrix A as $\|A\|_F = \sqrt{\mathrm{Tr}(AA^\dagger)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$. The hyperparameters for BFGS were chosen as $\epsilon_o = 10^{-6}$ and $M = 10000$ (see section 2.3.3).

We now briefly describe how this section is organized. We begin by analyzing the impact of recycling the Hessian on the measurement costs of ADAPT-VQE for multiple molecules, interatomic distances, and pools in section 4.1. In section 4.2, we seek to understand when our strategy works the best by analyzing the distance between the approximate and exact (inverse) Hessians, as well as the evolution of the latter. Finally, in section 4.3, we dive into specific optimizations to better understand the faster convergence of our method.

4.1. Measurement costs

In this subsection, we assess the impact of recycling the Hessian on the measurement costs of the optimization process. We consider the key cost to be the number of function evaluations. Evaluating the energy corresponds to one function evaluation, while evaluating an element of the gradient vector corresponds to two (see appendix A). The circuits for these two cases differ by a negligible number of gates, and the observable is always the Hamiltonian [45]. This means that each function evaluation requires measuring the same set of Pauli strings and implies a similar number of shots, up to differences in the variance of the expectation value of the energy in the state. Note that we can place a state-independent upper bound on the number of shots required for a given error [46]. Based on these arguments, we expect the total number of function evaluations to be a good figure for assessing costs.

Figure 1 shows the results for the H_6 molecule at various bond distances, using the QE and qubit pools. The first thing to note is that the error curves overlap for nearly all iterations, except for regimes of very high

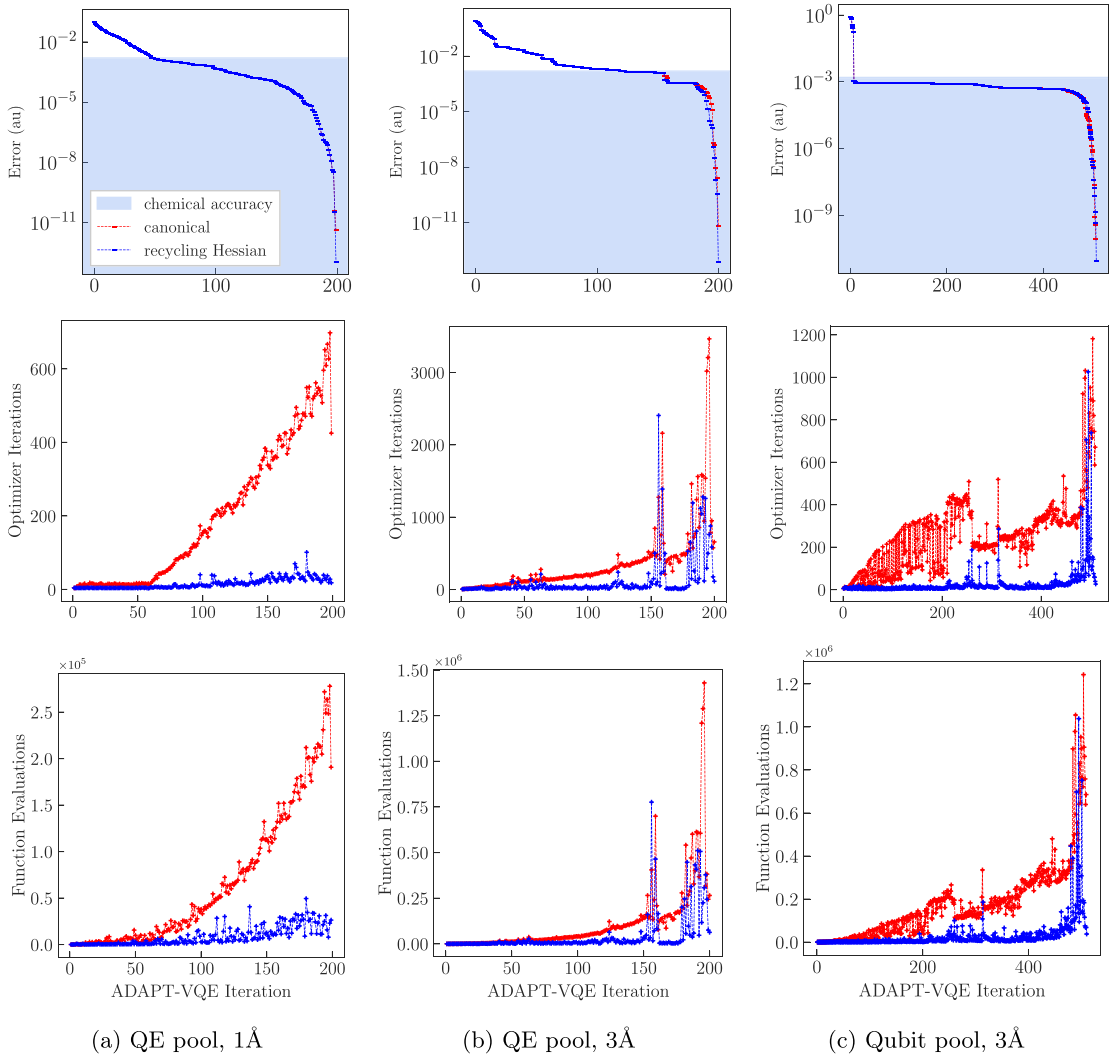


Figure 1. Comparison of the error (top row), total number of line searches (middle row), and total number of function evaluations (bottom row) per iteration of the ADAPT-VQE algorithm with and without Hessian recycling for H_6. We consider the QE pool for (a) equilibrium and (b) stretched geometries, as well as (c) the qubit pool for a stretched geometry. The red curve is often not visible in the error plots due to it lying below the blue one. The region shaded blue is the region of chemical accuracy, defined as an error below 1 kcal/mol.

accuracy where recycling the Hessian results in a lower absolute error. We do not expect this to be a benefit of our method, as this accuracy range is unlikely to be relevant in practice. Simulation data shows that up to such high accuracy regimes, the final ansätze are identical regardless of whether we recycle the Hessian.

Despite the matched accuracy, the number of optimizer iterations is remarkably different between the two methods. When the initial estimate for the inverse Hessian is the identity, the number of BFGS iterations clearly increases faster than linearly with the ADAPT-VQE iteration number; in contrast, when we recycle the Hessian, the number of line searches is roughly constant across a large number of ADAPT-VQE iterations, despite the considerable increase in the dimension of the parameter space. Since the second-order information is not reconstructed from scratch in each optimization, accounting for parameter correlations does not necessarily imply a superlinear number of optimizer iterations. In fact, the search direction for the very first line search at ADAPT-VQE iteration n is already equipped with information concerning correlations between $n-1$ parameters, represented by $(n-1)^2$ second derivatives. The only missing information concerns a linear number of second derivatives, describing the correlation between each parameter and the last.

Interestingly, the impact of recycling the Hessian for H_6 at 3 Å is more significant on qubit-ADAPT-VQE (figure 1(c)) than on QEB-ADAPT-VQE (figure 1(b)). With the qubit pool, the Hessian recycling protocol decreases the total number of function evaluations by 84%–20% more than the decrease with the QE pool. This suggests that the impact of the strategy is not only system- but also pool-dependent.

Table 1. Total measurement costs incurred by the QEB-ADAPT-VQE algorithm in the gradient measurement and VQE subroutines (steps 8 and 14 in algorithm 1), for the studied test cases. The costs are given as multipliers for the cost of a naive energy evaluation, so that unit cost implies $\mathcal{O}(N^4)$ measurements. For the gradient measurement step, we consider the worst-case cost under the leading measurement strategy for the QE pool ($8N$ per iteration, where N is the number of spin-orbitals/qubits) [24].

			Molecule					
			LiH		H_6		BeH_2	
			1.5Å	3Å	1Å	3Å	1.3Å	3Å
Step	Gradient Measurement		5.2×10^3	5.4×10^3	1.9×10^4	1.9×10^4	1.2×10^4	1.3×10^4
	VQE	Canonical	2.5×10^5	2.4×10^5	1.3×10^7	2.2×10^7	4.2×10^6	3.6×10^6
		Recycling Hessian (reduced to)	6.0×10^4 (24%)	3.2×10^4 (13%)	1.7×10^6 (13%)	7.9×10^6 (36%)	9.4×10^5 (22%)	5.6×10^5 (16%)

In general, for a fixed dimension, we expect a higher number of line searches to imply a higher number of function evaluations (see appendix B.1). This is confirmed by the bottom panels of figure 1, which show that the Hessian recycling strategy succeeds in producing a significant decrease in measurement costs.

In appendix C, we include additional plots for the LiH and BeH_2 molecules. In all cases, recycling the Hessian results in relevant savings in measurement costs. The savings are particularly notable for the H_6 molecule (especially at the equilibrium geometry). Among the three, this is the most difficult to simulate, requiring the most iterations and measurements. In all cases, the difference in costs seems to increase as the iterations advance. Thus, we can expect our method to become even more beneficial for larger systems.

Considering the same molecules at larger bond distances allows us to study the impact of our protocol as systems become more strongly correlated. While the savings in the number of line searches and measurements are maintained or even increased as we stretch LiH and BeH_2, they become less significant for H_6 at stretched geometries. Among the test cases we considered, this was the one where our strategy performed the worst. We investigate the causes in section 4.2. Despite this, recycling the Hessian still results in a reduction of the total number of function evaluations by 64% across the whole execution.

We finish this section with quantitative examples of the measurement cost reduction achieved by our method. Table 1 includes the costs of the VQE step with and without Hessian recycling, as well as of the gradient measurement step, through complete executions of QEB-ADAPT-VQE for various systems. Note that the latter step concerns the measurement of the gradients of operators generated by pool elements, not of operators in the ansatz (which are included in the costs for VQE). We do not consider any grouping strategies for the Hamiltonian, as they are beyond the scope of this work.

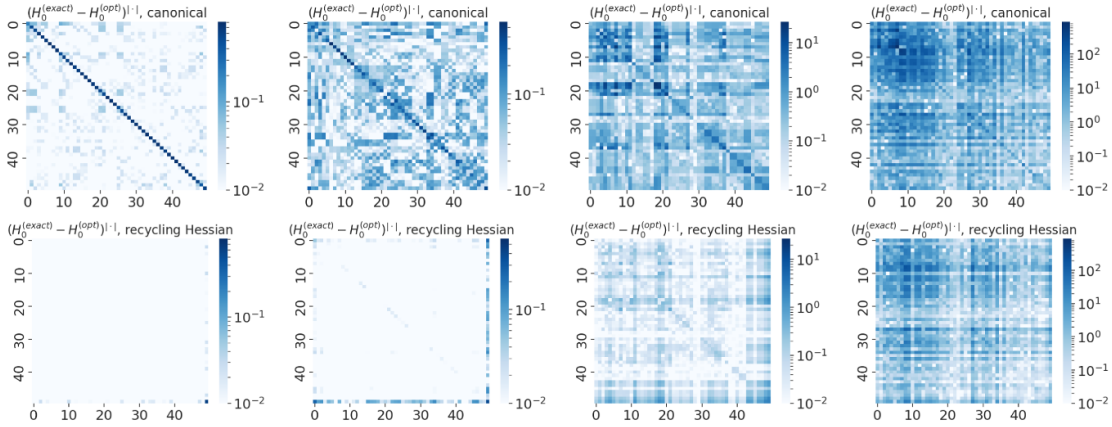
The numerical data shows that the measurement cost of the VQE subroutine is reduced by roughly one order of magnitude when our Hessian recycling strategy is employed. We additionally verify that, as predicted by our analysis in section 2.2.3, this subroutine is the bottleneck of the algorithm as far as measurement costs are concerned. In fact, the associated cost is always at least one order of magnitude higher than the cost of the gradient measurements, even when the Hessian recycling strategy is employed. As such, we confirm that our proposal addresses the most significant source of measurement costs of ADAPT-VQE, decreasing its *total* measurement costs by an order of magnitude (for the studied molecules—as discussed, we expect this decrease to become more significant for larger systems). For the majority of the molecules we considered, the reduction in the measurement costs of the optimization was in the 70%–90% range.

4.2. Distance to the exact Hessian

In this subsection, we analyze the distance between the approximate and exact inverse Hessians, $H^{(\mathrm{opt})}$ and $H^{(\mathrm{exact})}$, with and without Hessian recycling.

We begin with the heatmaps of figure 2, a visual representation of this distance at the beginning of the 50th optimization of ADAPT-VQE with the QE pool. We plot $(H_0^{(\mathrm{exact})} - H_0^{(\mathrm{opt})})^{|\cdot|}$, the element-wise distance between $H_0^{(\mathrm{opt})}$ and $H_0^{(\mathrm{exact})}$. We consider H_6 at four different bond distances, and we use the same color map for each bond distance. It should be noted that there is nothing unique about the 50th optimization. We chose this number due to it being high enough for the optimization process to be interesting, but not so high that the calculations become intractable or the heatmaps illegible. In appendix D we show that the behavior generalizes by presenting equivalent heatmaps for other iterations.

We note that due to how we initialize H_k in algorithm 3, the last row/column of the heatmaps is the same regardless of whether we recycle the Hessian or not. However, as expected, the inner second derivatives are better approximated by the previous iterations' derivatives than by the elements of the identity matrix. The difference is more significant for geometries near equilibrium.



(a) H_6 at 1Å (b) H_6 at 1.5Å (c) H_6 at 2Å (d) H_6 at 3Å

Figure 2. Heatmaps showing the difference between the initial approximate inverse Hessian in the optimization process and the initial exact inverse Hessian, with (bottom row) and without (top row) Hessian recycling, for the 50th iteration of QEB-ADAPT-VQE. The plots show the element-wise difference between these two matrices. Due to it being strongly correlated and the hardest to simulate, here we consider the test molecule H_6 at different interatomic distances.

Another interesting geometry-related trend can be observed in the upper heatmaps: for less correlated geometries (shorter bond distances), the approximation is the poorest for the diagonal elements [47]. As we increase the bond distance, there is a shift in behavior, as the magnitude of some off-diagonal elements in the difference matrix starts rivaling the magnitude of the diagonal elements. At 3 Å, the diagonal entries are no longer dominant. This is clearly symptomatic of a more complicated optimization: While for shorter bond distances the parameters can nearly be treated as uncorrelated, correlations between parameters play a bigger role for stretched geometries. Relatedly, the magnitude of the elements of the difference matrix changes from well below unity up to several hundred as we increase the bond distance.

While the heatmaps may help us gain intuition, the information they provide is limited, as they only concern one iteration. In figure 3, we plot the Frobenius distance between these same matrices for the first 100 iterations of QEB-ADAPT-VQE. We focus on the two extreme bond distances: 1 Å and 3 Å. As expected, the initial distance to the exact inverse Hessian is greater for the stretched bond distance, both when we recycle the Hessian and when we do not. Remarkably, the impact of recycling the Hessian on the initial distance is a similar multiplicative factor (close to 0.1) for both bond distances, despite the impact on costs being significantly larger for the 1 Å geometry (as we saw in figures 1(a) and (b)).

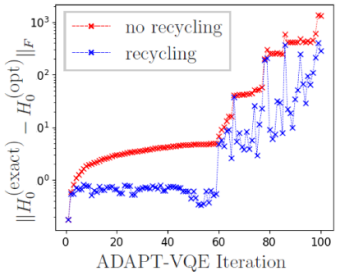
We hypothesize that in large part, the difference in performance is not due to the initial matrix being a poor approximation, but rather due to the optimization process being more difficult. If the initial point is farther away from a minimum, and the curvature around these two points is vastly different, the optimizer will require more iterations to move along the cost landscape, update the inverse Hessian, and reach the minimum. For such lengthier optimizations, the cost for the optimizer to move across the landscape is more likely to surpass the cost of approximating the second derivatives around the initial point. To test this hypothesis, we plot the distance between the exact initial and exact final inverse Hessians in figure 4.

The plot confirms that there is a significant increase in the distance between the initial and final inverse Hessians when the bond distance is increased. For H_6 at 1 Å, the distance varies between 10^{-2} and 10^{3} and only surpasses unity after 80 iterations. In contrast, at 3 Å, the distance varies between 10^{0} and 10^{4} and is above unity (and closer to the maximum value) throughout all the ADAPT-VQE iterations.

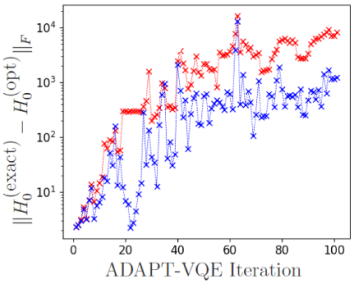
4.3. Evolution of the optimization

In the previous subsections, we focused on data from the beginning or end of each optimization. This allowed us to characterize the behavior of our protocol throughout complete runs of the ADAPT-VQE algorithm, and analyze how the dimension of the optimization and the molecule under study impact the costs.

However, as the protocol we propose is in fact a numerical optimization method, it is interesting to zoom in on the optimization process and investigate the impact of recycling the Hessian in quantitative aspects of it. Thus, in this section we delve into a particular optimization. We consider the 75th iteration of QEB-ADAPT-VQE for H_6 at equilibrium (1 Å) and stretched (3 Å) bond distances (the same systems we analyzed in the previous section). The corresponding optimization has 75 parameters. Once again, we note that there is nothing unique to the 75th iteration; we chose it because it is complex enough to be relevant, but

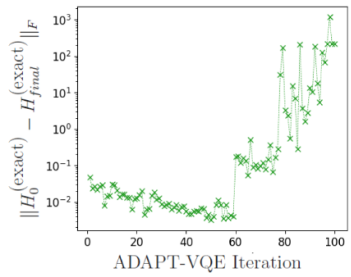


(a) H_6 at 1Å

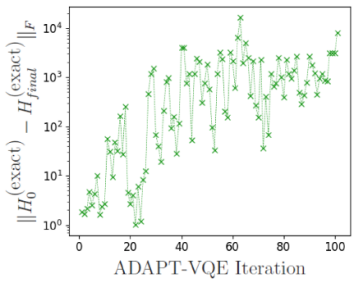


(b) H_6 at 3Å

Figure 3. Frobenius distance between the initial approximate inverse Hessian in the optimization process $H_0^{(\text{opt})}$ and the initial exact inverse Hessian $H_0^{(\text{exact})}$, with and without Hessian recycling, for the first hundred QEB-ADAPT-VQE iterations. We consider two cases: one for which the Hessian recycling works particularly well (H_6 at 1 Å) and one for which it does not (H_6 at 3 Å).



(a) H_6 at 1Å



(b) H_6 at 3Å

Figure 4. Frobenius distance between the exact initial $H_0^{(\text{exact})}$ and final $H_{\text{final}}^{(\text{exact})}$ inverse Hessians, for the first hundred QEB-ADAPT-VQE iterations. The systems are the same as those from figure 3.

not so much that calculating the relevant data becomes a computational challenge. In appendix E we consider a different iteration to show that the results generalize.

In the first column of figure 5 we observe that, as expected, the optimization requires significantly fewer iterations to converge for the equilibrium geometry. In line with previous results, recycling the Hessian results in similar final error for a lower number of iterations.

In the second column, we can see that the search direction (equation (8)) aligns much more quickly with the Newton direction (equation (6)) when the Hessian is recycled. We note that the initial distance to Newton's vector (which we plot as the value for iteration 0) is identical whether the Hessian is recycled or

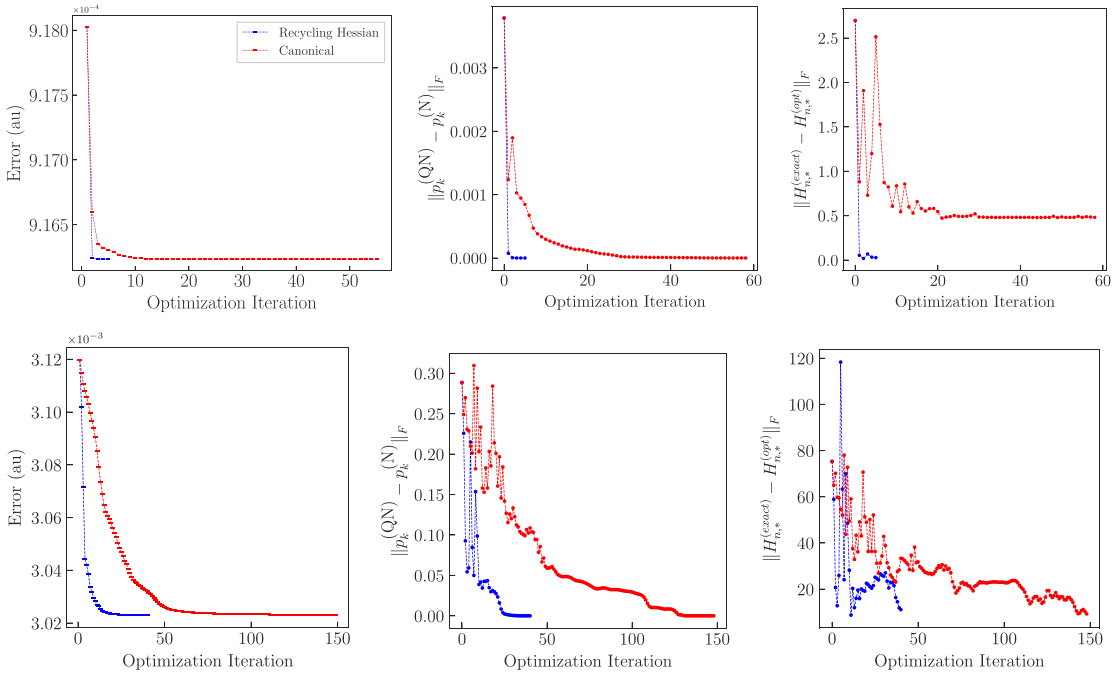


Figure 5. Evolution of the 75th QEB-ADAPT-VQE optimization with and without Hessian recycling, for the same two systems as in figure 3: H_6 at 1 Å (top row) and 3 Å (bottom row). The panels show (left to right) the error; the distance between the quasi-Newton $p_k^{(QN)}$ and Newton $p_k^{(N)}$ directions; and the distance between the last row of the approximate inverse Hessian $H^{(opt)}$ and the exact inverse Hessian $H^{(exact)}$.

not: the $n-1$ interior parameters were previously optimized and thus have nearly zero gradients, and step 3 in algorithm 3 is agnostic to any second-order information concerning the nth parameter. Despite being so similar initially, once the parameters start changing, the second-order information contained within the recycled Hessian becomes relevant and allows the direction to more quickly align with Newton's. This is remarkable: Unlike Newton's method, our optimization method never explicitly measures the Hessian, yet it aligns with the Newton search direction in a fraction of the iterations needed for the canonical BFGS implementation.

The third column of figure 5 shows the distance between the last row of the approximate and exact inverse Hessians. We denote the last row by $H_{n,*}$; recall that the matrix is symmetric, so that we have $H_{n,*} = H_{*,n}$. This is the vector of second-order derivatives involving the new parameter. For similar reasons to those explained in the discussion of figure 5(b), the initial distance, plotted at iteration 0, is identical regardless of whether the Hessian is recycled or not. Since the second-order derivatives being recycled only involve the parameters $\theta_1, \ldots, \theta_{n-1}$, naively, there seems to be no reason to expect that recycling the Hessian will lead to faster convergence of the last column/row (corresponding to θ_n). Surprisingly, this does happen; it seems that the initial curvature information concerning the interior parameters allows the optimizer to focus on exploring the new search direction, and enables a faster build up of information regarding correlations between the new parameter and old ones. Even more surprisingly, the distance is still higher for the canonical BFGS when convergence is reached, despite the larger number of iterations. This is particularly visible for the 1 Å geometry.

Finally, we focus on the convergence rate of the optimizations. The relevant quantities (see section 2.3.4) are plotted in figure 6. In the first column, we see that the difference between the approximate and exact Hessians along the search direction goes to zero in both optimizations when we recycle the Hessian, but this does not happen when we do not. Further, the step size (second column) saturates to unity after roughly half of the iterations when we recycle the Hessian, but oscillates instead of stabilizing when we do not. Together these results indicate that recycling the Hessian results in a superlinear convergence (equation (10)) that would otherwise not be achieved.

Finally, the third column of figure 6 confirms this. BFGS with Hessian recycling enjoys superlinear convergence, while canonical BFGS converges linearly with a convergence constant r close to 1—the worst possible scenario (see equation (9)). Such a convergence rate is expected of gradient descent in ill-conditioned problems, and constitutes an underwhelming performance for a quasi-Newton optimizer from which we hope to achieve superlinear convergence.

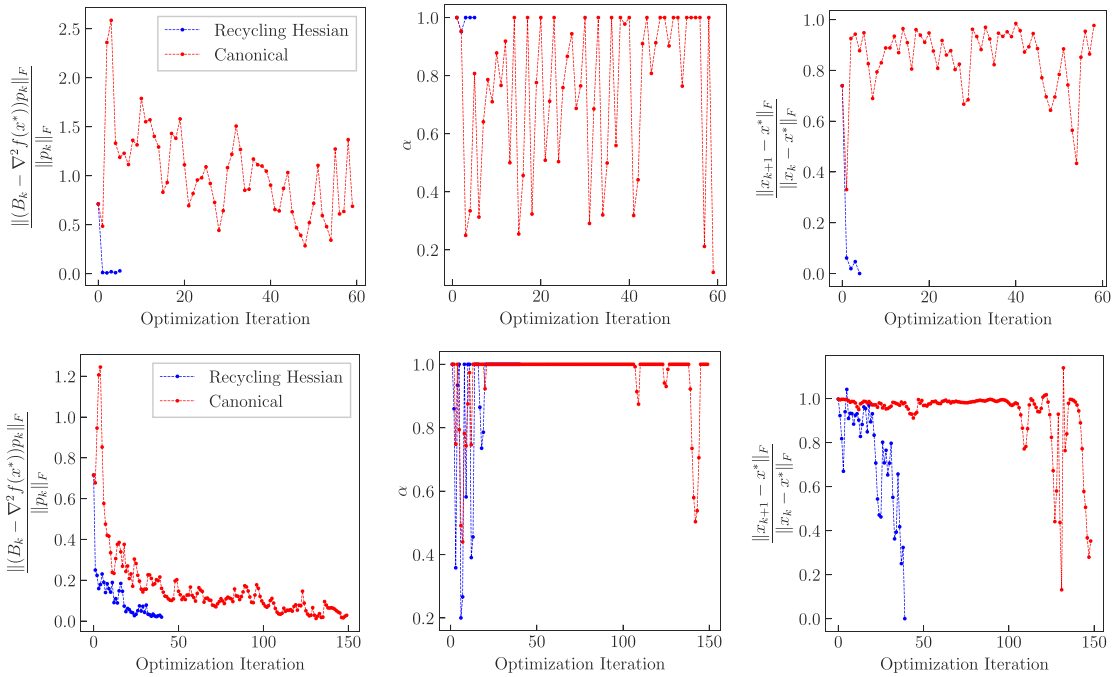


Figure 6. Relevant quantities for the study of the convergence rate of the 75th QEB-ADAPT-VQE optimization with and without Hessian recycling, for the same two systems as in figure 3: H_6 at 1 Å (top row) and 3 Å (bottom row). (a) The distance between the approximate and the exact Hessian along the search direction, (b) the step size α at the end of the line search, and (c) the ratio between successive errors are shown.

5. Conclusion

In this work, we proposed to tailor BFGS, one of the most popular optimizers for variational quantum algorithms, to ADAPT-VQE, one of the most popular such algorithms. In a typical implementation, each iteration of ADAPT-VQE performs an optimization where the state of the optimizer is initialized as if we possessed no knowledge about the curvature of the cost landscape, despite the fact that some knowledge was collected along the previous optimization. We develop a variant of the BFGS method that allows second-order information to naturally flow from one iteration to the next. By recycling the inverse Hessian matrix maintained by the optimizer, this protocol converges superlinearly even in situations where the canonical BFGS implementation does not, and achieves significant savings in the number of function evaluations required in each iteration of ADAPT-VQE. This specifically addresses the costs of the optimization process, which is the biggest source of measurement costs in such algorithms. In addition to decreasing the number of calls to the quantum computer in hardware implementations of adaptive VQAs, our strategy reduces the runtime of classically simulating them, thereby improving our ability to design and test these algorithms.

Since the impact of our Hessian recycling strategy seems to increase with the size of the system and the dimension of the optimization, we expect it to be even more beneficial for molecules for which performing classical simulations is infeasible.

While we focused on ADAPT-VQE for testing our proposed optimizer, our algorithm easily generalizes to other iterative state preparation protocols such as ADAPT-QAOA [11] or layerwise learning [15]. Freezing parameters or layering [6, 9] are also compatible with our strategy, requiring nothing more than a simple manipulation of the initial inverse Hessian (removing or adding subsets of rows and columns).

The noise-resilience of the proposed optimization algorithm is left as an open question. Adding realistic noise to the simulations implies an overhead in computational costs which leads to prohibitive simulation times for interesting molecules. Previous approaches to the noisy simulation of ADAPT-VQEs have averted this issue by growing the ansatz noiselessly and assessing the impact of noise in the final circuit exclusively [9, 48]. While this may provide insight regarding the near-term viability of such algorithms, it is evidently not applicable to the optimization process. Developing strategies for noisy simulation that address this issue is left for future work. We note that while there is a general expectation that optimizers designed to be robust in the presence of noise (such as SPSA [49]) will be better suited for experiments in quantum hardware, BFGS was shown in [50] to be among the best algorithms for noisy optimizations, with a performance comparable to SPSA.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/mafaldaramoa/ceo-adapt-vqe/tree/main.

Acknowledgments

We thank Raffaele Santagati and Matthias Degroote for their support and encouragement at the start of this Project, and Ernesto Galvão for helpful discussions. This work is in part financed by National Funds through the Portuguese funding agency, FCT—Fundação para a Ciência e a Tecnologia, within Project LA/P/0063/2020 (https://doi.org/10.54499/LA/P/0063/2020). MR acknowledges support from INESC TEC and FCT under PhD research scolarships 9575/BI-M-ED_B2/2022 and 2022.12333. BD respectively. NJM acknowledges support from the US Department of Energy (Grant No. DE-SC0024619). EB acknowledges support from the US Department of Energy (Grant No. DE-SC0022389). SEE acknowledges support by Wellcome Leap as part of the Quantum for Bio Program.

Appendix A. Evaluating gradients on quantum hardware

The gradient function ∇f which must be supplied to gradient-based optimization methods merits a discussion, since it is not evident that this information is available when the cost function is evaluated on a quantum computer.

We note that some implementations of BFGS (including SciPy's [42]) allow the user to bypass the construction of a gradient function by implementing it internally via finite difference (FD) methods. These numerical methods approximate derivatives from evaluations of the function at points which differ by small shifts, and can be applied to any function in a black-box fashion [51]. However, when the cost function is the expectation value of a (generic) quantum mechanical observable, it must be obtained by averaging over a finite number of samples, and thus it is inevitably noisy. In addition to this we have hardware limitations inherent to NISQ computers, such as miscalibrated rotation gates and other sources of noise. All of this hampers the task of gauging minute shifts in the function value, and the fact that the FD quotient must have a small denominator (because its magnitude is related to the magnitude of the error of the approximation) aggravates these problems. Expectably, BFGS with FD methods has been found to perform poorly in the presence of noise [52].

A realistic alternative to FD methods are parameter-shift rules (PSRs), proposed in [53] and extended in [54]. PSRs use a clever manipulation of analytical expressions to express the derivatives as linear combinations of measurable expectation values. Unlike FD formulas, which are generic numerical approximations, PSRs are analytical derivatives with a circuit-specific structure. When the qubit pool is used, the generators of the ansatz elements consist of a single Pauli string, whose gradient can be obtained from two energy measurements using the simplest PSRs. When the QE pool is used, more sophisticated techniques are required because each of the generators has three eigenvalues. In this case, the gradients can be measured using the fermionic PSRs proposed in [45]. For real wave functions (as we are concerned with), this will similarly imply the measurement of two expectation values, with the corresponding circuits having a negligible increase in circuit depth with respect to the energy measurement circuits.

Appendix B. BFGS optimizer

B.1. Line search subroutine

algorithm 2 in the main text requires a `line_search` subroutine to choose the next iterate. The task of this algorithm is to find the step size α which minimizes the function f along the search direction p_k,

$$\min_{\alpha>0} f(x_k + \alpha p_k)\,. \tag{B1}$$

We now briefly describe this subroutine.

Because finding the minimum with high accuracy might be unnecessarily costly, this minimization is often approximate and terminates when some reasonable conditions are met. A common choice are the Wolfe conditions,

$$f(x_k + \alpha_k p_k) \leqslant f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k, \tag{B2}$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geqslant c_2 \nabla f(x_k)^T p_k, \tag{B3}$$

where $0 < c_1 < c_2 < 1$. Example values for quasi-Newton methods are $c_1 = 10^{-4}$, $c_2 = 0.9$ [20]. Note that if p_k is a descent direction, as should be, then $\nabla f(x_k)^T p_k$ is negative.

Condition (B2), known as the *sufficient decrease* or *Armijo* condition, asserts that the decrease in the value of f is lower (and so higher in magnitude) than f's instantaneous rate of change along p_k at the initial point, weighed by c_1. This condition is evidently satisfied for small enough α_k, even if the decrease in the function value is negligible.

Condition (B3), known as the *curvature* condition, asserts that the derivative (along p_k) of the function at the new point is higher (and so lower in magnitude) than the one at the initial point, weighed by c_2. This bound is placed because the higher the magnitude of the rate of change of f, the higher the decrease in f we expect from further refining the step α_k.

Another important choice in a line search algorithm is how to initialize and vary α. One option is to start with a guess and increase it until we find a point which either satisfies the desired conditions or brackets points which do. In the latter case, we can backtrack, decreasing α until a valid point is found.

The cost of this subroutine depends on how many attempts it takes us to find an α that satisfies Wolfe's conditions (B2) and (B3), which depends on the shape of the cost function and the proximity to a minimum. Each iteration of the line search requires measuring the cost function and the gradient vector, which imply respectively 1 and $2n$ energy evaluations for an n-dimensional ADAPT-VQE optimization.

B.2. BFGS update rule

Another subroutine required by algorithm 2 is `update_h`, which updates the inverse Hessian H_k at the end of a line search. This update depends on the vectors s_k and y_k, obtained from the difference between the coefficient and gradient vectors, respectively, at the kth and $(k-1)$th iterations (see steps 8 and 9 of algorithm 2).

The BFGS update rule is obtained by enforcing three conditions:

1. The gradient of the new quadratic model for f (defined by a second-order Trotter expansion) matches the true gradient at the last two iterates.
2. H_{k+1} is symmetric and positive definite.
3. Among all matrices satisfying the above, H_{k+1} is the one which minimizes the distance to H_k, with respect to some norm.

These three conditions give rise to the BFGS update rule

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \tag{B4}$$

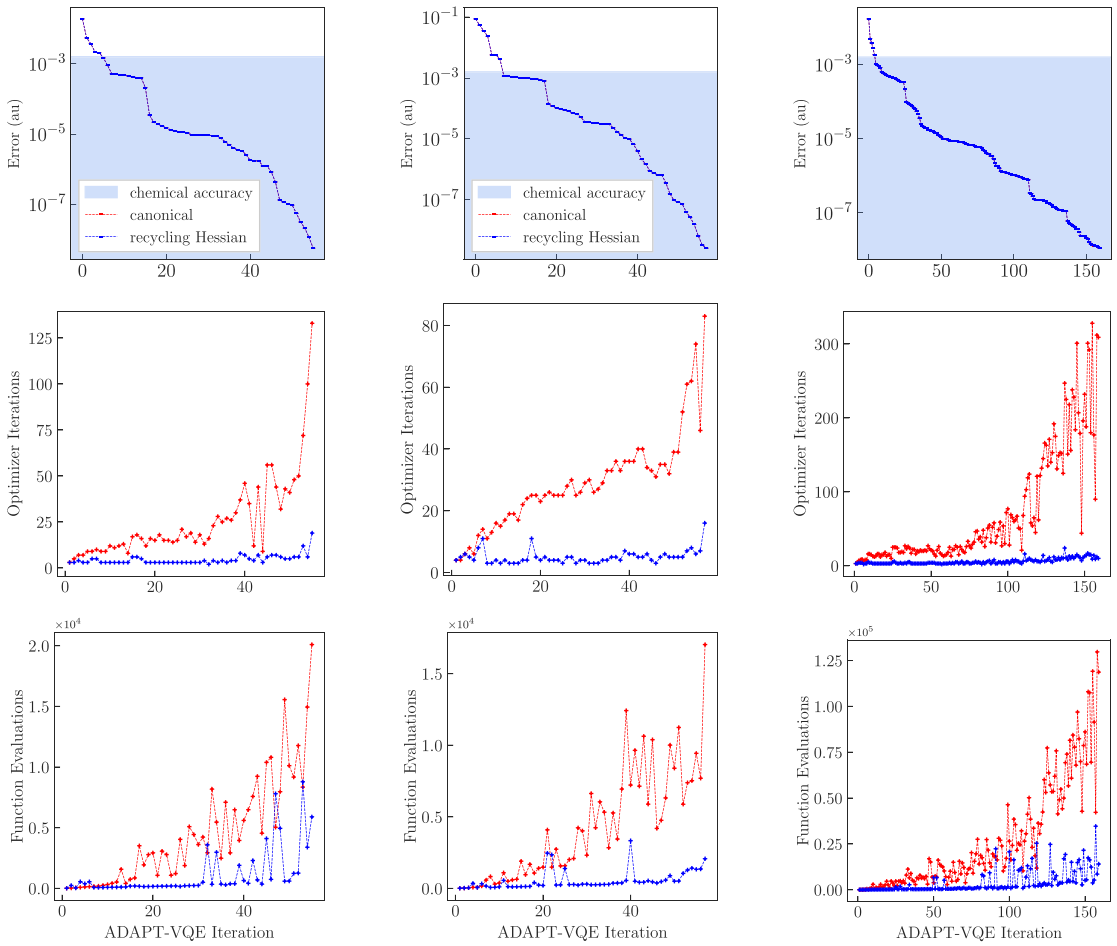
where $\rho_k = \frac{1}{y_k^T s_k}$. Conveniently, the BFGS method deals directly with the *inverse* Hessian, required to decide the next search direction via equation (8) of the main text. Other quasi-Newton formulas, such as DFP (proposed by Davidon in 1959 [55] and further analyzed by Fletcher and Powell [56]), handle the inversion posteriorly.

We note that the approximate inverse Hessian H (and the induced Hessian H^{-1}) will be positive definite even if the same is not true for the real Hessian (and its inverse).

B.3. Properties of BFGS

BFGS is a robust optimizer with good convergence [57] and self-correcting properties [58]. Despite never directly evaluating second-order derivatives, it enjoys a super-linear convergence rate and performs well in practice, reaching a minimum sufficiently fast for most practical purposes [17, 20]. Because of such features, this optimizer has become a popular choice for VQAs and is often used in practical implementations [3, 5, 21–23].

It has been proved that when implemented with Wolfe line searches (see appendix B.1.), this optimizer is globally convergent for convex functions, i.e. iterates will converge to a minimum regardless of the initial point [57]. In the context of numerical optimization, *global* is typically used to stress the independence of the convergence on the initial point [59]. Accordingly, local convergence properties are those that only hold when the initial point is close enough to a minimum. We note that in this context, the words 'local' and 'global' do not refer to the type of minimum. Globally convergent methods may converge to a local minimum and vice-versa.



(a) QE pool, 1.5Å (b) QE pool, 3Å (c) Qubit pool, 1Å

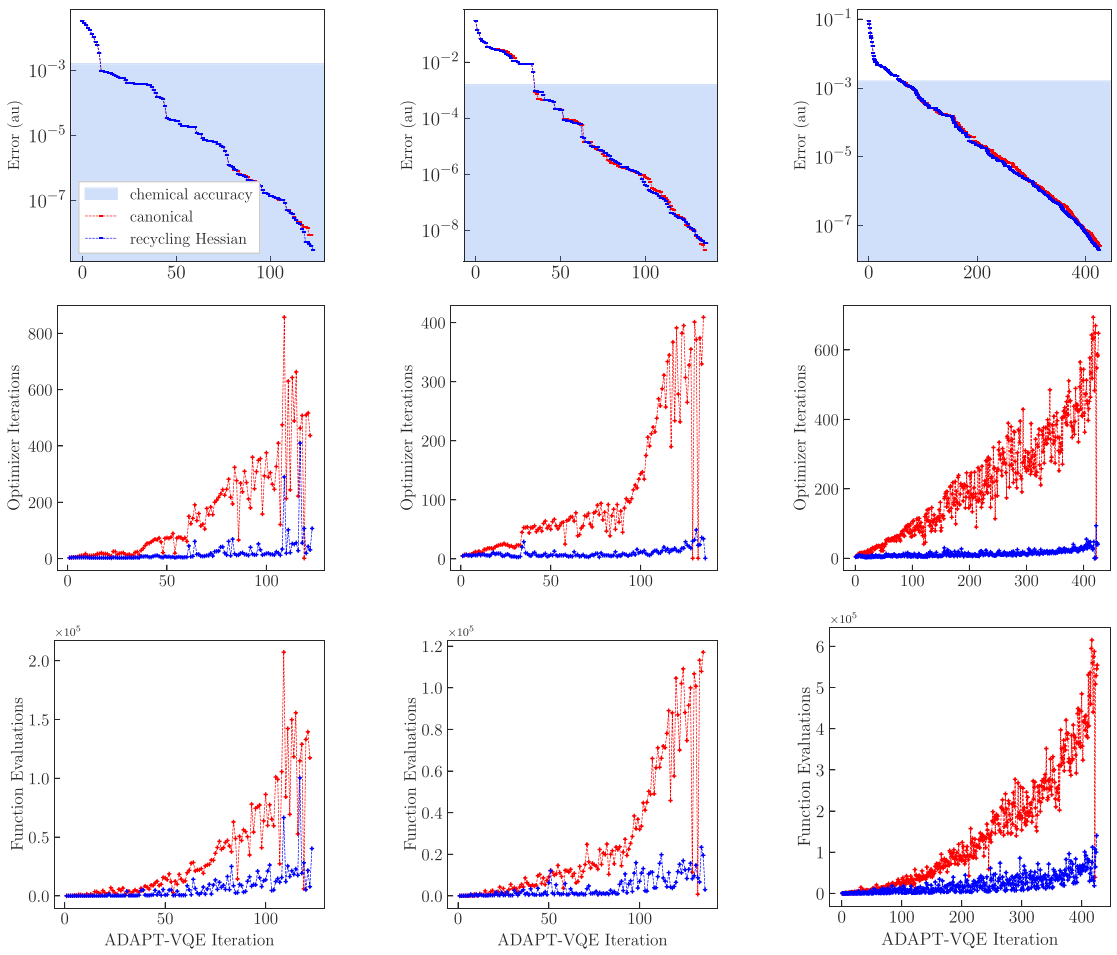
Figure 7. Equivalent plots to figure 1 for LiH.

Appendix C. Results for other molecules

In the main text, we showed that our Hessian recycling protocol results in a significant decrease in the measurement costs of ADAPT-VQE for H_6, and that this decrease tends to become more relevant as the system size increases. In this appendix, we show that these conclusions generalize to other molecules.

Figures 7 and 8 show the evolution of error, line searches, and measurement costs as the dimension of the ADAPT-VQE optimization grows. The molecules considered are respectively LiH and BeH_2. Once again, we consider both equilibrium and stretched geometries and two different pools.

We confirm that recycling the Hessian decreases the total measurement costs to a significant degree for all systems and pools.



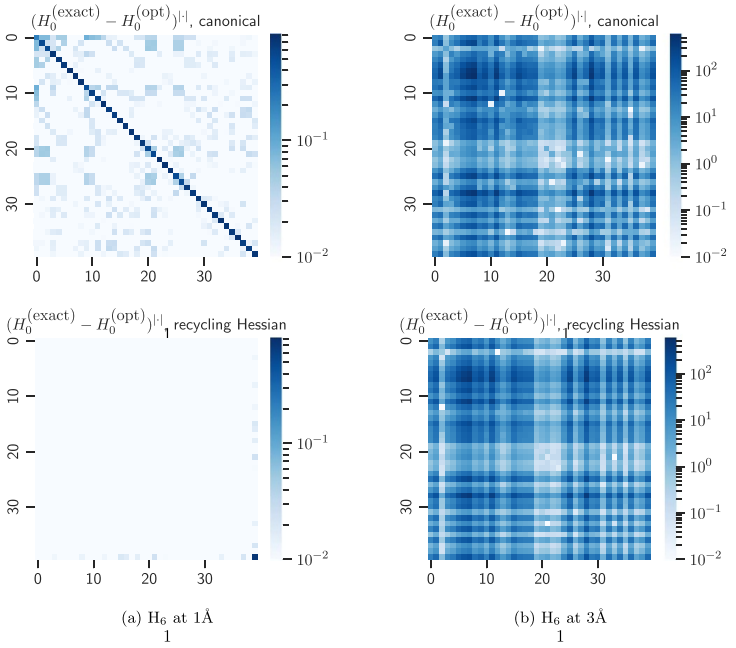
(a) QE pool, 1.3Å (b) QE pool, 3Å (c) Qubit pool, 2Å

Figure 8. Equivalent plots to figure 1 for BeH_2.

Appendix D. Heatmaps for other iterations

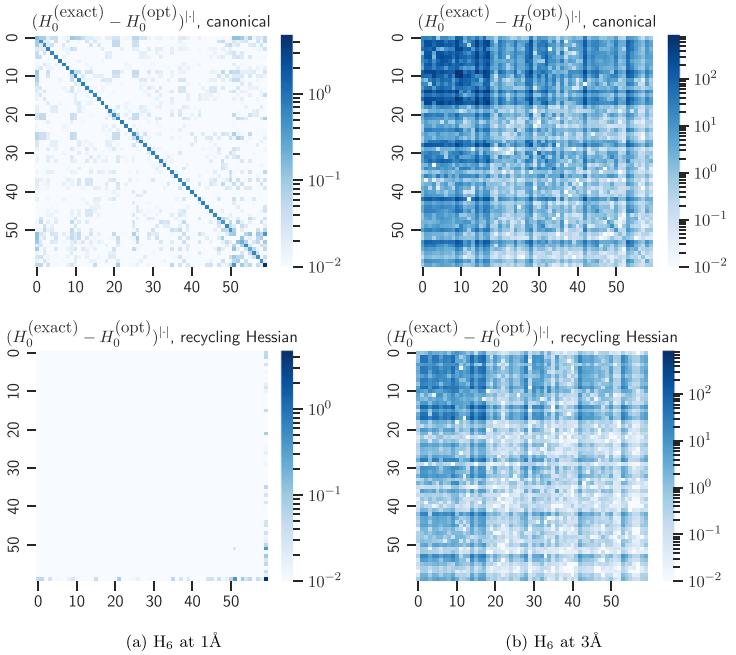
In order to confirm that the behavior showcased in figure 2 generalizes to other iterations, we consider equivalent heatmaps for other iterations. Figures 9 and 10 consider iterations 40 and 60, respectively.

As happened in the examples provided in the main text, we observe that the entries of the approximate Hessian are closer to the exact ones when we recycle the Hessian. While this happens for all test cases, once again we see that the impact of our protocol is more significant at the equilibrium geometry. Moreover, as happened in iteration 50, we can observe that the entries farther away from the true value are the diagonal ones for such a configuration, but that is no longer the case for the larger bond distance. The reasons behind these results were discussed in section 4.2.



(a) H_6 at 1Å (b) H_6 at 3Å

Figure 9. Heatmaps showing the difference between the initial approximate inverse Hessian in the optimization process and the initial exact inverse Hessian, with (bottom row) and without (top row) Hessian recycling, for the 40th iteration of QEB-ADAPT-VQE. The plots show the element-wise difference between these two matrices. We consider H_6 at equilibrium and stretched geometries.



(a) H_6 at 1Å (b) H_6 at 3Å

Figure 10. The equivalent of figure 9 for the 60th optimization.

Appendix E. Evolution of the 50th QEB-ADAPT-VQE iteration for H_6

In this appendix, we analyze a different optimization of QEB-ADAPT-VQE for H_6. Our purpose is to confirm that the results observed and discussed concerning the 75th optimization in section 4.3 were not fortuitous.

Figure 11 contains the same data depicted in figures 5 and 6, concerning H_6 at 1 Å; however, we now focus on the 50th iteration as opposed to the 75th one.

We confirm that the evolution of the optimization is similar. Recycling the Hessian speeds up the convergence of the energy, improves the alignment with Newton's direction, and brings the approximate

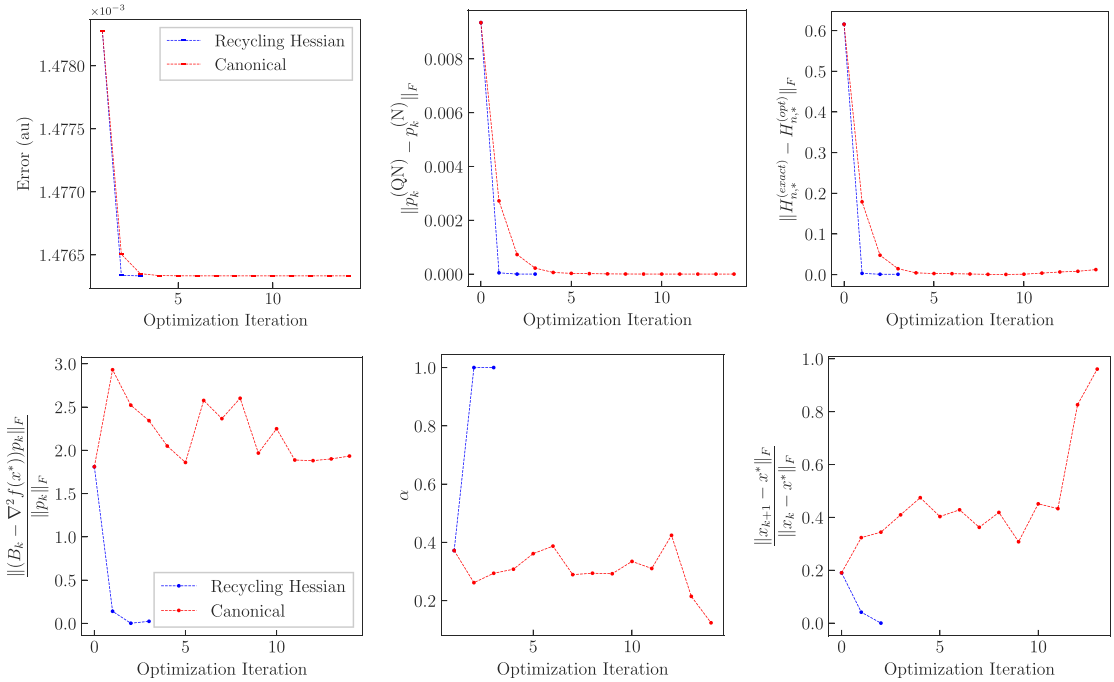


Figure 11. Relevant quantities for the study of the convergence rate of the 50th QEB-ADAPT-VQE optimization for H_6 at 1 Å.

Hessian closer to the exact one. Additionally, recycling the Hessian allows us to gather all conditions necessary for superlinear convergence, which we verify occurs in the last plot.

ORCID iDs

Mafalda Ramôa https://orcid.org/0000-0003-0218-7801
Luis Paulo Santos https://orcid.org/0000-0003-4466-1129
Edwin Barnes https://orcid.org/0000-0003-1666-9385
Sophia E Economou https://orcid.org/0000-0002-1939-5589

References

[1] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O'Brien J L 2014 A variational eigenvalue solver on a photonic quantum processor *Nat. Commun.* **5** 4231
[2] McClean J R, Boixo S, Smelyanskiy V N, Babbush R and Neven H 2018 Barren plateaus in quantum neural network training landscapes *Nat. Commun.* **9** 4812
[3] Grimsley H R, Economou S E, Barnes E and Mayhall N J 2019 An adaptive variational algorithm for exact molecular simulations on a quantum computer *Nat. Commun.* **10** 3007
[4] Tang H L, Shkolnikov V, Barron G S, Grimsley H R, Mayhall N J, Barnes E and Economou S E 2021 Qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor *PRX Quantum* **2** 020310
[5] Yordanov Y S, Armaos V, Barnes C H W and Arvidsson-Shukur D R M 2021 Qubit-excitation-based adaptive variational quantum eigensolver *Commun. Phys.* **4** 228
[6] Anastasiou P G, Chen Y, Mayhall N J, Barnes E and Economou S E 2022 Tetris-adapt-vqe: An adaptive algorithm that yields shallower, denser circuit ansätze (arXiv:2209.10562)
[7] Shkolnikov V, Mayhall N J, Economou S E and Barnes E 2021 Avoiding symmetry roadblocks and minimizing the measurement overhead of adaptive variational quantum eigensolvers (arXiv:2109.05340 [quant-ph])
[8] Bertels L W, Grimsley H R, Economou S E, Barnes E and Mayhall N J 2022 Symmetry breaking slows convergence of the adapt variational quantum eigensolver (arXiv:2207.03063)
[9] Long C K, Dalton K, Barnes C H W, Arvidsson-Shukur D R M and Mertig N 2023 Layering and subpool exploration for adaptive variational quantum eigensolvers: reducing circuit depth, runtime, and susceptibility to noise (arXiv:2308.11708 [quant-ph])
[10] Grimsley H R, Barron G S, Barnes E, Economou S E and Mayhall N J 2022 Adapt-vqe is insensitive to rough parameter landscapes and barren plateaus (arXiv:2204.07179)
[11] Zhu L, Tang H L, Barron G S, Calderon-Vargas F A, Mayhall N J, Barnes E and Economou S E 2020 An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer (arXiv:2005.10258)
[12] Chen Y, Zhu L, Liu C, Mayhall N J, Barnes E and Economou S E 2023 How much entanglement do quantum optimization algorithms require? (arXiv:2205.12283 [quant-ph])
[13] Sridhar V K, Chen Y, Gard B, Barnes E and Economou S E 2023 Adapt-qaoa with a classically inspired initial state (arXiv:2310.09694 [quant-ph])

[14] Yanakiev N, Mertig N, Long C K and Arvidsson-Shukur D R M 2023 Dynamic-adapt-qaoa: An algorithm with shallow and noise-resilient circuits (arXiv:2309.00047 [quant-ph])
[15] Skolik A, McClean J R, Mohseni M, van der Smagt P and Leib M 2020 Layerwise learning for quantum neural networks (arXiv:2006.14904 [quant-ph])
[16] Lee X, Saito Y, Cai D and Asai N 2021 Parameters fixing strategy for quantum approximate optimization algorithm *2021 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) (https://doi.org/10.1109/qce52317.2021.00016)
[17] Dennis J J E Jr and Moré J J 1977 Quasi-newton methods, motivation and theory *SIAM Rev.* **19** 46
[18] Dennis J E and Schnabel R B 1996 *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Society for Industrial and Applied Mathematics) (https://doi.org/10.1137/1.9781611971200)
[19] Fletcher R 1987 *Practical Methods of Optimization* 2nd edn (Wiley)
[20] Nocedal J and Wright S J 1999 *Numerical Optimization* (Springer) (https://doi.org/10.1007/b98874)
[21] Guerreschi G G and Smelyanskiy M 2017 Practical optimization for hybrid quantum-classical algorithms (arXiv:1701.01450 [quant-ph])
[22] Lotshaw P C, Humble T S, Herrman R, Ostrowski J and Siopsis G 2021 Empirical performance bounds for quantum approximate optimization *Quantum Inf. Process.* **20** 403
[23] Mbeng G B, Fazio R and Santoro G 2019 Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes (arXiv:1906.08948 [quant-ph])
[24] Anastasiou P G, Mayhall N J, Barnes E and Economou S E 2023 How to really measure operator gradients in adapt-vqe (arXiv:2306.03227 [quant-ph])
[25] McClean J R, Romero J, Babbush R and Aspuru-Guzik A 2016 The theory of variational hybrid quantum-classical algorithms *New J. Phys.* **18** 023023
[26] Bartlett R J and Musia M 2007 Coupled-cluster theory in quantum chemistry *Rev. Mod. Phys.* **79** 291
[27] Romero J, Babbush R, McClean J R, Hempel C, Love P J and Aspuru-Guzik A 2018 Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz *Quantum Sci. Technol.* **4** 014008
[28] Grimsley H R, Barron G S, Barnes E, Economou S E and Mayhall N J 2023 Adaptive, problem-tailored variational quantum eigensolver mitigates rough parameter landscapes and barren plateaus *npj Quantum Inf.* **9** 19
[29] Jordan P and Wigner E P 1993 Über das paulische Äquivalenzverbot *The Collected Works of Eugene Paul Wigner* (Springer) pp 109–29
[30] Nielsen M A and Chuang I L 2012 *Quantum Computation and Quantum Information* (Cambridge University Press) (https://doi.org/10.1017/cbo9780511976667)
[31] Yordanov Y S, Arvidsson-Shukur D R M and Barnes C H W 2020 Efficient quantum circuits for quantum computational chemistry *Phys. Rev.* A **102** 062612
[32] Gokhale P, Angiuli O, Ding Y, Gui K, Tomesh T, Suchara M, Martonosi M and Chong F T 2020 Optimization of simultaneous measurement for variational quantum eigensolver applications *2020 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) (https://doi.org/10.1109/qce49297.2020.00054)
[33] Yen T-C, Verteletskyi V and Izmaylov A F 2020 Measuring all compatible operators in one series of single-qubit measurements using unitary transformations *J. Chem. Theory Comput.* **16** 2400
[34] Tilly J *et al* 2022 The variational quantum eigensolver: A review of methods and best practices *Phys. Rep.* **986** 1
[35] Napp J 2020 Variational quantum algorithms and geometry *Quantum Views* **4** 37
[36] Fletcher R 1982 Second order corrections for non-differentiable optimization *Lecture Notes in Mathematics* (Springer) pp 85–114
[37] More J J and Sorensen D C 1982 Newton's method (https://doi.org/10.2172/5326201)
[38] Broyden C G 1970 The convergence of a class of double-rank minimization algorithms *IMA J. Appl. Math.* **6** 222
[39] Fletcher R 1970 A new approach to variable metric algorithms *Comput. J.* **13** 317
[40] Goldfarb D 1970 A family of variable-metric methods derived by variational means *Math. Comput.* **24** 23
[41] Shanno D F 1970 Conditioning of quasi-newton methods for function minimization *Math. Comput.* **24** 647
[42] Virtanen P *et al* 2020 SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python *Nat. Methods* **17** 261
[43] et al J R M 2020 OpenFermion: the electronic structure package for quantum computers *Quantum Sci. Technol.* **5** 034014
[44] Sun Q *et al* 2018 Pyscf: the python-based simulations of chemistry framework *WIREs Comput. Mol. Sci.* **8** e1340
[45] Kottmann J S, Anand A and Aspuru-Guzik A 2021 A feasible approach for automatically differentiable unitary coupled-cluster on quantum computers *Chem. Sci.* **12** 3497
[46] Wecker D, Hastings M B and Troyer M 2015 Progress towards practical quantum variational algorithms *Phys. Rev.* A **92** 042303
[47] One could hypothesize that this is a matter of scaling. However, the errors in the diagonal remain dominant even if we multiply the identity matrix by a scalar factor tuned to minimize the distance to the true values.
[48] Dalton K, Long C K, Yordanov Y S, Smith C G, Barnes C H W, Mertig N and Arvidsson-Shukur D R M 2022 Variational quantum chemistry requires gate-error probabilities below the fault-tolerance threshold (arXiv:2211.04505)
[49] Spall J 1992 Multivariate stochastic approximation using a simultaneous perturbation gradient approximation *IEEE Trans. Autom. Control* **37** 332–41
[50] Pellow-Jarman A, Sinayskiy I, Pillay A and Petruccione F 2021 A comparison of various classical optimizers for a variational quantum linear solver *Quantum Inf. Process.* **20** 202
[51] Grossmann C, Roos H-G and Stynes M 2007 *Numerical Treatment of Partial Differential Equations* (Springer) (https://doi.org/10.1007/978-3-540-71584-9)
[52] Lavrijsen W, Tudor A, Muller J, Iancu C and de Jong W 2020 Classical optimizers for noisy intermediate-scale quantum devices *2020 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) (https://doi.org/10.1109/qce49297.2020.00041)
[53] Mitarai K, Negoro M, Kitagawa M and Fujii K 2018 Quantum circuit learning *Phys. Rev.* A **98** 032309
[54] Schuld M, Bergholm V, Gogolin C, Izaac J and Killoran N 2019 Evaluating analytic gradients on quantum hardware *Phys. Rev.* A **99** 032331
[55] Davidon W 1959 *Variable Metric Method for Minimization* (Office of Scientific and Technical Information (OSTI)) (https://doi.org/10.2172/4222000)
[56] Fletcher R and Powell M J D 1963 A rapidly convergent descent method for minimization *Comput. J.* **6** 163

[57] Powell M Atomic Energy Research Establishment (Harwell, England), Atomic Energy Research Establishment (Harwell, England). Computer Science and Systems Division 1975 *Some Global Convergence Properties of a Variable Metric Algorithm for Minimization Without Exact Line Searches* (*CSS*) (UKAEA, Harwell Atomic Energy Research Establishment) (available at: https://books.google.com/books?id = JDLgPQAACAAJ)
[58] Nocedal J 1992 Theory of algorithms for unconstrained optimization *Acta Numer.* **1** 199
[59] Sriperumbudur B K and Lanckriet G R G 2012 A proof of convergence of the concave-convex procedure using zangwill's theory *Neural Comput.* **24** 1391