Qubit Control Simulation: Final Report

Sam Schwartz

MQST Solid State Quantum Computing Laboratory
University of California, Los Angeles

June 9, 2025

I. ABSTRACT

In this lab, we explore the implementation and optimization of a variational quantum eigensolver (VQE) [1] for solving H_2 ground states using parametric compilation on Rigetti's quantum architecture. We first developed the framework for Hamiltonian generation using Google Quantum AI's OpenFermion package. We then designed a basic VQE implementation to accurately run on quantum simulation. Lastly, we optimized this implementation and investigated the effect of parametric compilation for execution on actual quantum hardware. Through this process, we were able to design an optimized, parametrically compiled VQE that is both highly accurate (mean squared error of .019303) and efficient (3.78x reduction in quantum processor time, 14.46x reduction in total runtime) on Rigetti's 'Ankaa-3' quantum computer.

II. INTRODUCTION

Quantum Computers have matured to the point where noisy-intermediate-scale quantum (NISQ) devices can effectively solve classically hard problems provided that these algorithms are implemented with device limitations in mind. Hybrid quantum-classical protocols such as the VQE have therefore emerged as leading candidates for near-term chemistry and molecular materials simulations. In this lab we focus on superconducting qubit hardware supplied by Rigetti Computing, utilizing its Quil/PyQuil programming framework and the 82-qubit Ankaa-3 architecture to estimate ground-state energies of molecular hydrogen (H_2). Our contributions are as follows: (i) automated generation of H_2 Hamiltonians and energy approximations using OpenFermion, (ii) construction of a compact, hardware-efficient UCCSD ansatz ran on an optimized VQE solution instantiation, and (iii) demonstration of significant runtime reduction via parametric compilation. The resulting implementation achieves chemically accurate energies on a quantum virtual machine and, when ported to Ankaa-3, delivers a 3.78x speed-up in QPU time and a 14.46x reduction total runtime relative to non-parametric problem executions, all while maintaining a low mean-squared error of 0.019303 Hartree.

III. THEORY

A. Variational Quantum Eigensolvers

1. Overview

The Variational Quantum Eigensolver (VQE) was first proposed by Peruzzo et al. in 2014 [1]. The variational principle states that given a Hamiltonian $\hat{H}$ and a wave function $|\psi\rangle$, the associated ground state energy E_0 is bounded by equation 1. The goal of a VQE instantiation is to find some parametrization of $|\psi\rangle$ that minimizes the expectation value of $\hat{H}$. Mathematically, this can be re-defined as finding an optimal approximation of the eigenvector $|\psi\rangle$ of $\hat{H}$ that corresponds to the minimum eigenvalue E_0 [2].

$$E_0 \leq \frac{\langle\psi|\,\hat{H}\,|\psi\rangle}{\langle\psi|\psi\rangle} \tag{1}$$

If we model equation 1 as an optimization problem, we can execute on a quantum computer. We do this through the use of a parametrized unitary ansatz $U(\theta)$ where $\theta \in (-\pi, \pi]$ or $(0, \pi]$ depending on implementation. If we implement $\hat{H}$ in terms of a weighted sum of Pauli operators over N qubits, we can define the VQE optimization problem/cost function as follows (where $|\hat{0}\rangle = |0\rangle^{\otimes N}$) [2]:

$$E_{opt} = \min_{\theta} \langle\hat{0}|\, U^{\dagger}(\theta)\hat{H}U(\theta)\,|\hat{0}\rangle \tag{2}$$

2. Problem Workflow

The pipeline towards solving a VQE problem using a quantum computer can be broken down into 4 main components: Hamiltonian Construction/Encoding, Ansatz/State Preparation, Measurement, and Parameter

Optimization [3]. A mathematically based description of this process, with labels for both classical and quantum subroutines, is shown in Figure 1 from Tilly et al.

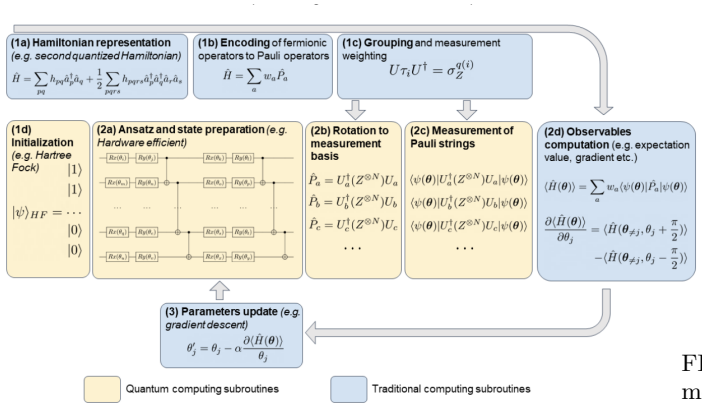

FIG. 1. VQE Workflow diagram from Tilley et al. [2]

B. OpenFermion Hamiltonian Construction

OpenFermion is a Python library "for compiling and analyzing quantum algorithms to simulate fermionic systems, including quantum chemistry" from Google Quantum AI [4]. We can leverage this package's ability to computationally build Hamiltonians out of fermionic operators comprised for H_2 molecules of varying bond lengths. From there, we can map these Hamiltonians to weighted sums of Pauli operators using the Jordan-Wigner transformation, and lastly convert these to PyQuil PauliSum operators for experimental use [5]. The Hamiltonian derived for a bond length of .75 Angstroms is shown in Figure 2. Clearly, this is a 4-qubit Hamiltonian, which is not the fully reduced form for a H_2 molecule but still allows us to optimize effectively over just one parameter θ. In Figure 3 we analyze the accuracy of OpenFermion's ground state energy calculations by plotting them against published results from Helgaker et al. [6] across a range of bond lengths.

```
(-0.10973055606700657+0j)*I +
(0.1698845202794036+0j)*Z0 +
(0.1698845202794036+0j)*Z1 +
(-0.2188630678121963+0j)*Z2 +
(-0.21886306781219633+0j)*Z3 +
(0.16821198673715718+0j)*Z0*Z1 +
(0.045442884144326186+0j)*Y0*X1*X2*Y3 +
(-0.045442884144326186+0j)*Y0*Y1*X2*X3 +
(-0.045442884144326186+0j)*X0*X1*Y2*Y3 +
(0.045442884144326186+0j)*X0*Y1*Y2*X3 +
(0.12005143072546032+0j)*Z0*Z2 +
(0.16549431486978650+0j)*Z0*Z3 +
(0.1654943148697865+0j)*Z1*Z2 +
(0.12005143072546032+0j)*Z1*Z3 +
(0.17395378776494125+0j)*Z2*Z3
```

FIG. 2. OpenFermion Hamiltonian for H_2 at .75 Å [6]

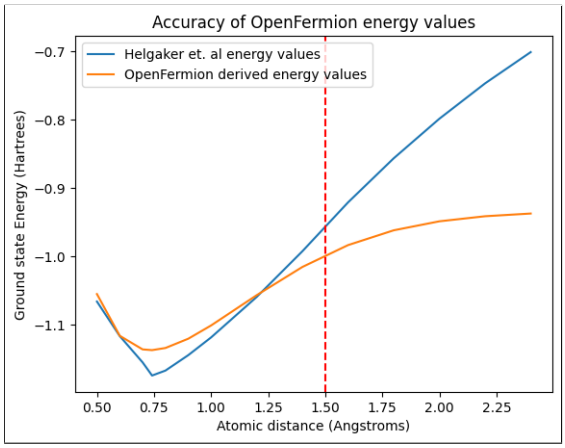

FIG. 3. Comparison of OpenFermion calculated to experimentally derived ground state energies [6]

Using the visual in Figure 3 as a baseline, we can see that the accuracy of the OpenFermion ground state energy estimates significantly tails off around 1.5 Angstroms. This is confirmed through comparing the mean squared error of the plot as a whole (.00969) to that of the plot up to 1.5 Angstroms (.00048). With this in mind, we use a bond length 1.5 Angstroms as an experimental upper bound on energy calculations.

C. Unitary Coupled-Clusters Singles & Doubles (UCCSD) Ansatz

The ground state wave function (and corresponding ground state energy) of a molecular Hamiltonian $\hat{H}$ can be approximated systematically by unitary coupled-cluster (UCC) theory [7]. Because of its' unitary nature, this allows us to implement a parametrized UCC ansatz efficiently on a quantum computer. We define the wavefunction of UCCSD ansatz as follows (where $\hat{\theta}$ is a vector of all circuit parameters θ, $|\phi\rangle = |110...0\rangle$ which is equivalent to the Hartree-Fock state, and $\hat{T}$ is the cluster excitation operator defined further in equation **??**) [8].:

$$|\psi(\hat{\theta})\rangle = e^{\hat{T}-\hat{T}^\dagger}\,|\phi\rangle \tag{3}$$

Since it is computationally infeasible to include all possible excitations of $\hat{T}$, it is common (and often times sufficient) to truncate it at the second/doubles level (hence the name Unitary Coupled-Clusters Singles & Doubles). In the below equation, the $t_{i(j)}^{a(b)}$ terms correspond to excitation from occupied orbitals into virtual orbitals [8].

$$\hat{T} = \hat{T}_1 + \hat{T}_2 = \sum_{i\in \text{occ}} \sum_{a\in \text{virt}} t_i^a\, \hat{a}_a^\dagger \hat{a}_i + \sum_{i,j\in \text{occ}} \sum_{a,b\in \text{virt}} t_{ij}^{ab}\, \hat{a}_a^\dagger \hat{a}_b^\dagger \hat{a}_i \hat{a}_j. \tag{4}$$

One can then apply the Jordan-Wigner transformation to map each excitation operator to a string of Pauli terms

that allow us to express equation 3 as a quantum circuit which will compile down into a series of single qubit parametrized rotation gates and 2-qubit control gates dependent on the native-gate set of the device [8].

D. Parametric Compilation

Hybrid quantum-classical algorithms like VQE generally work by running many repetitions of the same circuit while only changing the parametrized value(s) θ corresponding to angle(s) of basis rotation(s). Using standard compilation procedures, every single instance of a parameter changing/updating will cause the compiler to need to recompile the circuit with an updated θ value. This poses a significant time and energy cost to the processor. Parametric Compilation works by compiling the circuit once with a symbolic variable θ_{sym} that points towards the memory address of the θ values $\in \hat{\theta}$. The pointer is updated during run-time, and no recompilation or rerunning of code is needed. Parametric compilation in PyQuil can be easily implemented as shown in [9]. Theoretically, parametric compilation will significantly reduce the compile and run-time of a circuit while improving its' fidelity since calibrations are only done once [10].

IV. EXPERIMENTAL PROCEDURES

A. Rigetti Quantum Suite

In this project we utilize the power of the Rigetti quantum suite in order to execute experiments both on quantum simulators and actual quantum devices. For simulation experiments, I used Rigetti's '4q-qvm' (4 qubit quantum virtual machine) to simulate compiled Quil programs. These simulation results/processes are designated as QVM experiments throughout this paper. For experiments done on quantum hardware, I used Rigetti's 'Ankaa-3' quantum processing unit (QPU). Ankaa-3 is Rigetti's premier 82 qubit device; its' lattice and some important device specs can be seen in Figure 4. Use of Ankaa-3 costs \$0.000002 per microsecond, in total these experiments cost $\approx \$4,700$ of QPU credits.

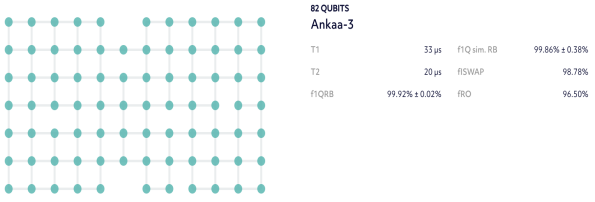

FIG. 4. Rigetti's Ankaa-3 QPU device specs [11]

B. Codebase Overview

In this section I provide a overview of the programming framework used to execute QVM & QPU experiments. The full code for this experiment in Jupyter notebook form is published on a Github repository **here**. There are 2 separate notebook files, one that compiles the circuit parametrically and one that does not. Outside of this difference in compilation approach and some minor accompanying changes these notebook files are essentially interchangeable/identical and will be treated as such.

First, we prepare our Hamiltonian as outlined in section III B. The method

```
build_Hamiltonian(bond_length)
```

returns the OpenFermion calculated ground energy and the Hamiltonian (as a PauliSum operator) for a specified bond length. From there, we loop over each Hamiltonian term, apply basis changes as needed so that we are only measuring along the Z-axis, and compile each term as it's own small circuit.

Due to the relative simplicity of the solution for the H_2 state and the robustness of the UCCSD ansatz, our parameter vector $\hat{\theta}$ only needs to have one term in it (i.e each circuit execution only needs to optimize over one parameter) to fully span the Bloch sphere. The larger the size of the parameter grid (possible values $\hat{\theta} = \theta$ can take), the smaller the 'steps' between points circumnavigating the Bloch sphere. Next, we loop over the points in the parameter grid in batches, execute each compiled circuit (re-compilation is needed if not using parametric compilation), and use our classical energy evaluation function

```
batch_energy_eval(parameter_batch)
```

to find the ground state energy for each value in the parameter grid. We can then simply take the minimum ground state energy found from the parameter grid loop and use that as our estimated, minimized E_0 value.

C. Experimental Constants Interlude

Listed below are experimental values that remained constant (unless otherwise noted) throughout all experimental runs:

- Circuit depth ("depth") = 3
- Experiment shots ("shots") = 100
- Batch Size for circuit execution ("batch_size") = 25
- Individual bond length experiments ("bond_len") = .75 Angstroms

- Bond length loop experiments ("bond_lengths") = a linear space of 20 evenly spaced points between .1 and 1.5 Angstroms.

- Parameter grid ("theta_grid") = a linear space of 31 evenly spaced points between 0 and π

D. Experimental Procedures

1. *QVM Experiments*

On the Rigetti QVM we ran 2 experiments (one parametric and one static/non-parametric), looping over the bond lengths in order to see how well our VQE instantiation models the true ground state energy derived from the Open Fermion Hamiltonian. These experiments were executed 10 times each, with average values plotted in Figures 5 and 6. There is also a table comparing the two approaches (V A).

2. *QPU Experiments*

On the Rigetti QPU, we ran 5 experiments. The first two were focused on attempting to find an optimal size of the parameter grid. These experiments were executed 5 times each at .75 Angstroms for a size 10,20,30,40,50, and 60 parameter grid. Figures 7 and 8 show both the average mean squared error and QPU processing time for each data point. The next two experiments compared an unoptimized/naive run of VQE to our fully optimized implementation. The two main changes made to this optimized version are implementing the UCCSD ansatz (as opposed to layers or uniquely parametrized RY gates followed by entangling the 4 qubits together through CNOT gates) and manually selecting the $[0, 1, 7, 8]$ qubits to be used as the ansatz (as opposed to what the compiler selects). The difference between the unoptimized and optimized runs of the VQE are shown in Figures 9 and 10. These experiments were executed 5 times each, with the error bars representing the maximum and minimum value found at each individual point and the plotted point being the mean ground state energy value at each bond length across the 5 experimental runs. The last experiment done compares an optimized parametric run on the QPU to an optimized non-parametric/static run on the QPU. Due to reasons that will be elaborated on further in the paper, only one non-parametric experimental run was executed. The parametric experiments were executed 5 times each with the same error-bar/plotted point logic as the prior experiment. There is also a table that compares these two approaches in more detail (V B).

V. EXPERIMENTAL DATA

A. QVM Experimental Data

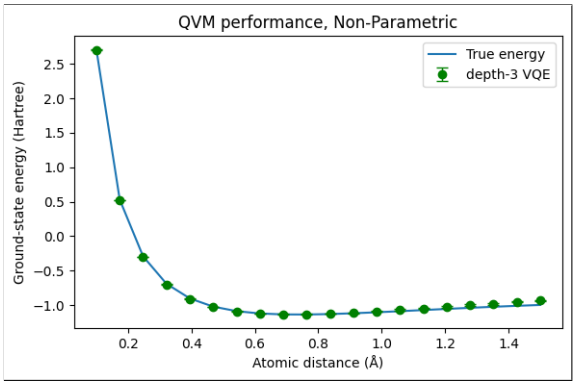

FIG. 5. QVM ground state energy estimates (static)

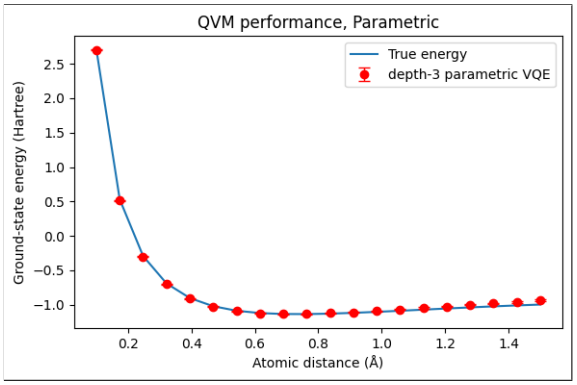

FIG. 6. QVM ground state energy estimates

QVM Compilation Comparison	Static	Parametric
Mean Squared Error	.000690	.0000674
Avg. CPU time per point	.737s	1.80s
Avg. CPU time per iteration	15.5s	36.1s
Avg. wall time per point	12.4s	1m 7s
Avg. wall time per iteration	4m 8s	22m 19s

TABLE I. QVM Compilation Comparison

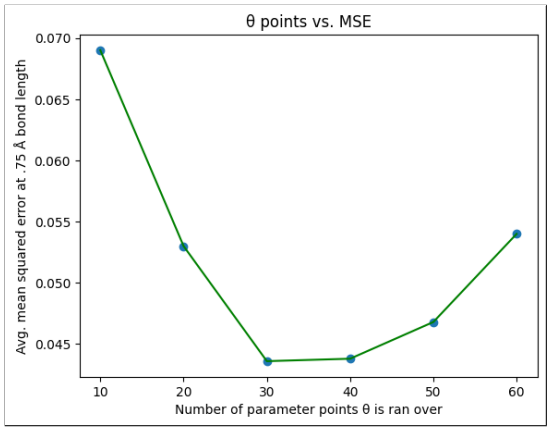

FIG. 7. Amount of points in parameter grid vs. MSE

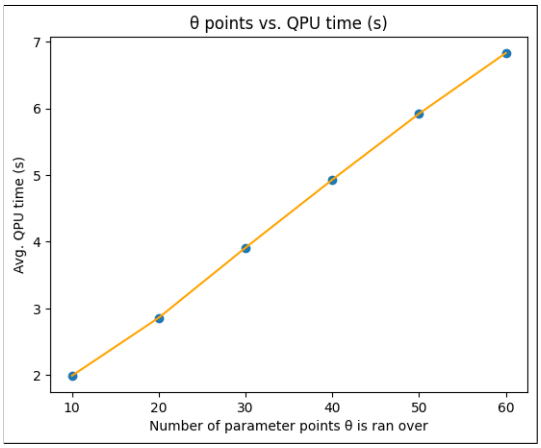

FIG. 8. Amount of points in parameter grid vs. time

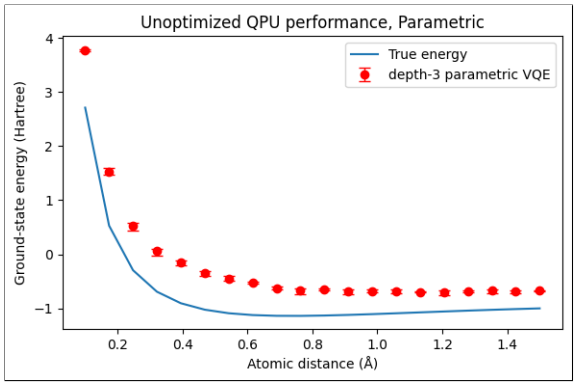

FIG. 9. Unoptimized QPU ground state energy estimates

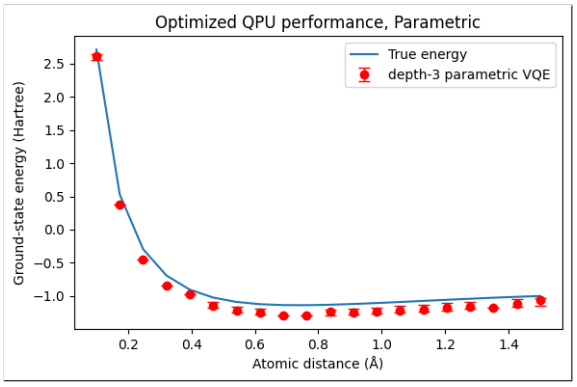

FIG. 10. Optimized QPU ground state energy estimates

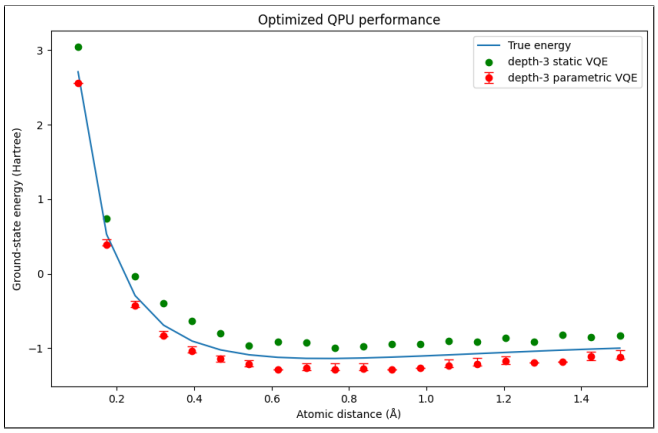

FIG. 11. Parametric vs. Static QPU ground energy estimates

QPU Compilation Comparison	Static*	Parametric
Mean Squared Error	.042560	.019303
Avg. QPU time per point	3.60s	13.62s
Avg. QPU time per iteration	1m 12s	4m 32s
Avg. wall time per point	21.7s	5m 18s
Avg. wall time per iteration	7m 13s	1hr 46m

TABLE II. QPU Compilation Comparison

B. QPU Experimental Data

VI. ANALYSIS

1. *QVM Experiments*

Figures 5 & 6 show the results of our ground state energy calculations of both static and parametric compilation over bond lengths between .1 and 1.5 Angstroms for two essentially identical VQE instantiations. The y axes of these graphs are energy (in Hartrees) and the x axes are the atomic distance of the bond lengths in Angstroms. We can visually inspect that these both

very accurately predict the OpenFermion derived ground state energies (represented by the blue curve) for these values, although we can see they start to tail off a bit at around 1.4 Angstroms. This slight disconnect can be explained by referencing Figure 3, where we can see that the OpenFermion calculations actually start to underestimate the scientifically derived ground state energies. This excellent solution accuracy verifies the validity of our VQE solution approach.

Table (V A) compares the two approaches in terms of mean squared error, CPU time, and wall time (equivalent to total run time of the process). The accuracy in terms of mean squared error between these two approaches can be considered essentially equivalent which is to be expected. The power of using parametric compilation on CPUs can be seen in the average run time data. There is a 2.33x reduction in CPU time and a 5.4x reduction in total run time when using parametric compilation. The magnitude of this reduction in time will only increase for larger, more complex problems. We can clearly see the power of implementing parametric compilation here, as there is a significant decrease in run time and computational power needed to arrive at just as strong as a VQE result.

2. *QPU Experiments*

The first experiments executed on the QPU were focused on analyzing the effect of the size of the parameter grid on both the accuracy (7) and time complexity (8) of a VQE instantiation for solving the ground state energy at .75 Å. These experiments were executed at parameter grid sizes of 10,20,30,40,50 and 60. Figure 7 plots parameter grid size vs. the average mean squared error over 5 runs. I expected this data to resemble an exponential decay, but it strangely starts to increase in error from grid sizes of 40 to 60. One of the reasons these results are so confounding is due to the fact that a larger parameter grid should theoretically only increase the probability of an accurate ground state energy estimation as there are just more values of θ to analyze the circuit on. Nonetheless, if we take this data at face value it motivates our choice for experimentation on a parameter grid of size ≈ 30 (we choose 31 to so that there is a point roughly every .1 sized step between 0 and π). The data in Figure 8 makes much more sense, as it shows the linear scaling in average QPU run time (s) over the parameter grid size. This makes sense, as adding parameter points linearly increases the amount of circuit executions needed, thereby linearly increasing the processing time required of the QPU.

The next experiment executed on the QPU shows the effect of applying the UCCSD ansatz and manual optimal qubit selection to our VQE run - showcasing the improvement in accuracy present when doing so. The axes & true energy plot are the same as in our QVM experiments. The error bars represent the maximum and minimum values over 5 experiments, where the point plotted is the mean. The unoptimized run is shown in Figure 9 and the optimized run is shown in Figure 10. While the shape of the unoptimized performance is actually fairly accurate, it consistently converges too far too large of a minimum ground state energy, especially for small bond lengths. The implementation of the UCCSD ansatz and optimal qubit quadrant allows our VQE performance to be much stronger - to the point that actually converges to consistently slightly too low of a ground state energy across the space of bond lengths. Not only can we clearly see the improvement visually, but there is 18x reduction in the average mean squared error (from .339 to .019) between the optimized and unoptimized implementations. I still think there is a lot of room for improvement here, as the optimal qubit patch I found was arrived at through essentially guessing and checking patches around the corners of the lattice - there is significant room for future work both in terms of experimentally deriving the optimal qubit patches to use on the Ankaa-3 chip as well as testing the extent to which qubit selection improves algorithmic success.

The final experiment executed on the QPU compares the effect of running VQE for both of the optimized parametric and optimized static cases. The plot (Figure 11 follows the same convention as above, with no error bars being present on the static compilation run due to cost and time constraints allowing for only one experimental run (5 runs were done for the parametric case). Table (V B) provides mean squared error data amd time analysis. Visually, the plot is very interesting, as it seems to imply a roughly equivalent performance in terms of accuracy but with the static compilation underestimating the ground state energy and the parametric compilation overestimating the energy. They almost perfectly bound the true energy curve between them, oddly enough. The table confirms this relative accuracy equivalence, although the parametric does perform a bit better the lack of data for the static runs does not allow us to confidently conclude that it is superior in terms of accurate performance. What is clear, is the massive reduction in QPU and especially total run time. This is particularly significant, as mentioned before, every microsecond increases the financial cost of operation. The data tells us that there is a 3.78x reduction in QPU time and a massive 14.46x reduction in total run time. Not only do these results make a strong case for the necessity of parametric compilation for a problem of this size, but the scale of improvements should only increase as the problem size grows. This experiment not only cements that parametric compilation is a necessity when running quantum optimization problems, but also encourages the potential for effect quantum optimization runs on NISQ hardware.

3. Next Steps & Open Questions

Both the fiscal and time constraints of conducting this research as well as the multi-faceted nature of studying optimization problems have led to a significant amount of potential extensions to this project that, unfortunately, I have not been able to explore. The results of this experiment could potentially be improved through conducting optimization experiments over the number of experimental shots and circuit depth (i.e. # of ansatz layers). Ideally, I would have been able to explore conducting my own gate calibrations and comparing them to Rigetti's native device calibrations as well. There is also a significant amount of untapped potential in executing a robust, exhaustive grid search over the Ankaa-3 lattice (Figure 4) in order to find the optimal qubit patch for our ansatz.

The constrained nature of this research has also led to multiple questions that remain unanswered through the conclusion of this project. Some of these questions are listed below, and I encourage any interested readers to explore possible answers theoretically or experimentally.

- Why does the static VQE implementation consistently overestimate the true ground state energy whereas the parametric VQE implementation consistently underestimates it (Figure 11)? Is this just small sample noise?
- Is there an explanation behind the behavior in Figure 7 that implies a more robust parameter grid leads to worse performance?
- How does this VQE implementation scale to larger, more complex molecular Hamiltonians (such as LiH or H_2O)?

VII. CONCLUSION

In summary, we have shown that a thoughtfully engineered VQE pipeline - combining OpenFermion-based Hamiltonian synthesis, a compact UCCSD ansatz, and parametric circuit compilation - can provide an accurate and efficient estimation of H_2 ground state energies. Across both simulation and Ankaa-3 QPU execution, our implementation reproduces chemically accurate ground state energies for H_2 (mean squared error = .019303) while drastically cutting down on QPU time by 3.78x and total run time by 14.46x relative to a naive approach. These results can be further improved through more robust optimization experiments, as well as ideal qubit selection. Also, the Rigetti Ankaa-3 chip is subject to significant amounts of noise, and has qubits with frequently very low T_1 and T_2^* times. The black-box QPU execution is a significant potential source of error. These results highlight parametric compilation as an essential tool for scaling and implementing variational algorithms and suggest that, with continued improvements in qubit selection and device calibration, similar speed-accuracy trade-offs can be realized for larger molecular systems and more complex quantum-chemistry benchmarks.

VIII. ACKNOWLEDGMENTS

I would like thank my fellow UCLA MQST students who partook in the Qubit Control Simulations module (Lorenzo Ballerio, Keshavi Charde, and Runzhao Guo) for their help and insight throughout this process. I would also like to thank the QuElements team, specifically Amy Brown and Huo Chen for their immense help throughout this project in terms of both answering questions and interfacing with Rigetti. I'd also like to thank our 'Lab Czar' Mohamad Niknam for his guidance throughout this project. A special thank you goes out to the Rigetti team as a whole and F.D.C Willard.

[1] A. P. et al., A variational eigenvalue solver on a photonic quantum processor, Nature Communications, 5 (2014).
[2] J. T. et al., The variational quantum eigensolver: a review of methods and best practices, Physics Reports, 986 (2022).
[3] D. Boneh and W. Zeng, Cse 269q, lecture 9 notes, stanford Elements of Quantum Computer Programming course notes.
[4] OpenFermion, Quantum simulation of electronic structure, OpenFermion guide tutorials (2025).
[5] P. Jordan and E. Wigner, Über das paulische Äquivalenzverbot, Z. Physik 47 (1928).
[6] T. Helgaker, P. Jorgensen, and J. Olsen, *Molecular Electronic-Structure Theory* (Wiley, 2013).
[7] P. O. et al., Scalable quantum simulation of molecular energies, Phys. Rev. X 6, 031007 (2016).
[8] D. A. Federov, Y. Alexeev, S. K. Gray, and M. J. Otton, Unitary selective coupled-cluster method, Quantum 6, 703 (2022).
[9] PyQuil Docs, *Parametric Compilation Examples*.
[10] R. S. Smith, E. C. Peterson, M. G. Skilbeck, and E. J. Davis, An open-source, industrial-strength optimizing compiler for quantum programs, Quantum Sci. Technol. 5 (2020).
[11] Rigetti QCS, *Ankaa-3 Lattice*.