A Variational Quantum Eigensolver in Q#

Christopher Kang, Robert Minneker, Julian Roth
University of Washington, Seattle
Final Project, CSE 490Q - Svore

June 22, 2020

Abstract

Q# is a domain specific language used to express quantum algorithms. The provided Quantum Chemistry Library features three different Hamiltonian simulation methods: Trotterization, Qubitization, and linear combinations of unitaries. We implement a fourth method, the Variational Quantum Eigensolver [1]. Our implementation is compatible with existing libraries and Microsoft's Broombridge Hamiltonian format. Last, we create a conversion program for the PSI4 format to Broombridge.

Introduction & Problem Statement

The primary near-term application of quantum computers is quantum chemistry. Quantum computers are thought to be able to tackle quantum chemistry problems exponentially faster than classical computers. A primary problem is the identification of ground state energies - these simulations are crucial for synthesizing catalysts, modeling chemical reactions, and identifying molecular structure. Speedups in quantum chemistry would enable the simulation of molecules that would be intractable classically [2, 3, 4].

Traditionally, quantum chemistry problems are presented as simulating a Hamiltonian, with a specific goal of identifying the Hamiltonian's eigenstate and eigenvalue. By finding an eigenstate and eigenvalue, the simulated molecule's ground energy and configuration can be identified (and thus its interactions with other molecules). Current classical methods suffer when attempting to approximate the quantum waveform, with complexity of $O(n^k), k > 3$ depending on the method of simulation [5]. Classical methods may only consider valence shells to reduce the complexity of the problem, but the identification of ground states of Hamiltonians remains an NP problem.

In response, different quantum approaches have been developed to simulate Hamiltonians. Trotterization relies upon exponentiating and segmenting the non-unitary Hamiltonian matrix to produce a series of unitaries that can be applied to a qubit array, representing electrons. The application of these unitaries yield an estimate to the energy level of the represented state. Trotterization scales in $O(\frac{m^{3/2}t^{3/2}}{\sqrt{\epsilon}})$ time with respect to the number of terms (m), time step (t), and the precision (ϵ) [6]. However, Trotterization requires the use of Quantum Phase Estimation, which can take millions of gates [7]. The circuit depth poses an issue as modern day quantum computers are far from mathematical ideals. Coined "Noisy Intermediate-Scale Quantum" (NISQ) devices [8], near-term devices will have short decoherence times and limited available qubits.

VQEs

Variational quantum eigensolvers (VQEs) are classical-quantum hybrid methods for identifying the eigenvector and eigenvalue of an arbitrary matrix. In order to find the eigenvalue of the input matrix, we begin by recognizing that an arbitrary Hamiltonian can be represented as:

$$H = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \sigma^i_\alpha \sigma^j_\beta \tag{1}$$

Thus, we may find the expectation value of H, $\langle H_{|\psi\rangle}\rangle$, by identifying the individual expectation values of the constituent Pauli matrices:

$$\langle H_{|\psi\rangle}\rangle = \sum_{i\alpha} h^i_\alpha \langle\sigma^i_\alpha\rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle\sigma^i_\alpha \sigma^j_\beta\rangle \tag{2}$$

We may now find the individual expectation values and sum them classically. This allows for much shallower circuits in comparison to simulating complex exponentiated matrices and utilizing quantum phase estimation to extract expectation values, as used with the Trotterization simulation method. The tradeoff, however, is that

the ansatz preparation step must be repeated to calculate the expectation value of each term.

Currently, many simulations have been performed on current quantum computers with VQEs due to their minimal circuit requirements [9]. VQEs pose a resource-efficient way to tackle quantum chemistry problems given immediately realizable NISQ devices.

Methods

Q# already provides an encoding scheme ('Broombridge') that represents the Hamiltonians provided by NWChem [10], an open source chemistry simulation module. Because Hamiltonians are expressed as a series of creation and annihilation operators, a transformation is needed to map the Hamiltonian to an analogy suitable for quantum computation. These terms are transformed into a series of Pauli applications with the Jordan-Wigner Transformation. This existing framework left multiple problems to tackle:

1. Compatibility of VQE code with Broombridge and other library datatypes
2. Conversion of Jordan-Wigner terms to individual Pauli terms
3. Conversion of other computational chemistry formats into Broombridge
4. Parametrized creation of arbitrary ansatz (or state of the molecule) and optimization of the parameters

To convert the Jordan-Wigner terms and accept arbitrary Hamiltonians, we created an alternative conversion system of the transformation: instead of sequentially applying the unitaries as implemented in the Quantum Chemistry Library, we return an array of the individual unitaries to apply. Through *partial application*, a feature of Q#, operations can have their targets selectively given. The unitary array allows helper methods to return expectation values of the individual Pauli terms, which are multiplied by coefficients and summed.

To convert other formats into Broombridge, we further developed a Python script which takes PSI4 integral data and rewrites it in the Broombridge format. By connecting PSI4 to Broombridge, users can utilize PSI4 files on any of the 3 existing simulation methods (and provided VQE code).

Resource Estimation

VQE is a randomized algorithm, that is the running time will depend on the Jordan Wigner Encoding of our initial state and the success rate of the state preparation algorithm. Both ResourcesEstimator, which we used for the gate counts, and QCTraceSimulator, which we used for gate depth, do not actually simulate the state of the Qubits. For this reason we need to provide them with the success probability of our state preparation algorithm. We provide two metrics: an optimistic (opt) estimation assuming that state preparation will always succeed on the first run and a second estimation where we have estimated the success rate of the state preparation algorithm using a Maximum Likelihood Estimator (MLE). After 100 trials we get a MLE of the success rate of the state preparation algorithm of 0.826 for H2_2 and 0.138 for H4.

We provide the qubit width of our algorithm (which is not random), and the count and depth of CNOT, T and QubitClifford gates. The reason we will also provide count and depth of QubitClifford gates is that depending on the Jordan Wigner Encoding of our molecule, it's possible to restrict our circuit solely to Pauli gates for both the Hamiltonian simulation and the state preparation. In this case the count and depth of our CNOT and T gates will be zero (e.g. H2O).

Gate counts and depths

	H2_2	H4
CNOT (opt)	3,967,040	67,965,920
T (opt)	2,550,240	44,629,200
QubitClifford (opt)	1,965,920	15,392,960
CNOT (MLE)	4,788,378	497,941,030
T (MLE)	3,078,243	326,968,425
QubitClifford (MLE)	2,334,684	112,194,980
Width	11	17

Results

In order to assess the accuracy of the method in comparison to Trotterization, the experimental VQE's energy estimates were compared over two molecules: H4 and H2_2. Each molecule was simulated 20 times on a cloud instance with 32 threads. The initial ansatz state given to the VQE was the NWChem suggested ground state.

H2_2 (n=20)

	Trotter	VQE	SCF
μ	-2.21505	-2.13518	-2.14567
σ	0.01544	0.03787	-

H4 (n=20)

	Trotter	VQE	SCF
μ	-1.78781	-1.51103	-1.55858
σ	0.39271	0.07087	-

The initial results are promising, showing energy estimates similar to available polynomial time algorithms, like Hartree-Fock methods, also known as self-consistent field (SCF) methods. In addition, note that the standard deviation of the VQE trials was reasonable given the number of trials and is comparable to the standard deviation of the Trotter approximation.

More accurate classical algorithms often rely upon a Full-Configuration Interactions (FCI) appraoch. These algorithms are more computationally complex and produce an interval of their prediction for a specific probability. When comparing to the FCI predictions, there are some concerns over inaccurate energy states:

	Lower	FCI μ	Upper
H2_2	-2.31609	-2.21609	-2.11609
H4	-2.01552	-1.91552	-1.81552

It is apparent that the Trotter simulation is closer to the FCI μ for both molecules. However the VQE predictions are within the FCI bounds for H2_2 and both predictions are outside FCI bounds for H4. This means that VQEs will produce results at least comparable to modern classical algorithms with the potential for speedups, depending on the cost of preparation of ansatzes. These results imply that with further work on the improvement of VQEs, better estimates could be obtained.

As for performance, the circuit width scales linearly with the number of simulated orbitals. This means that the VQE is comparable to Trotterization with the qubit requirement, actually taking one fewer qubit (as phase estimation is not needed).

Psi4 to Broombridge Conversion

We can express an arbitrary Hamiltonian by

$$H = \sum_{i,j} \sum_{\sigma \in \{\uparrow,\downarrow\}} h_{ij} a^\dagger_{i,\sigma} a_{j,\sigma} + \frac{1}{2} \sum_{i,j,k,l} \sum_{\sigma,\rho \in \{\uparrow,\downarrow\}} h_{ijkl} a^\dagger_{i,\sigma} a^\dagger_{k,\rho} a_{l,\rho} a_{j,\sigma} \quad (3)$$

where

$$h_{ij} = \int dx \psi^*_i(x) \Big(\frac{1}{2} \nabla^2 + \sum_A \frac{Z_A}{|x - x_A|} \Big) \psi_j(x) \quad (4)$$

is the one electron integral over spin orbitals i and j, and

$$h_{ijkl} = \int \int dx^2 \psi^*_i(x_1) \psi_j(x_1) \frac{1}{|x_1 - x_2|} \psi^*_k(x_2) \psi_l(x_2) \quad (5)$$

is the two electron integral over spin orbitals i, j, k and l.

We can compute these integrals using classical algorithms. For this we use an open-source quantum chemistry library called Psi4, which we can use to compute the required integrals, energies and energy offsets for some given molecule. Before using this data in Q#, we need to convert it to the Broombridge format. Broombridge is human-readable and can be parsed by Q#'s chemistry library to return the Jordan-Wigner Encoding of our molecule and its Hamiltonian. This encoding is directly used in the VQE algorithm to maintain compatibility with existing libraries.

Note: our current version of the conversion algorithm doesn't yet add initial state suggestions. This is an optional field in Broombridge and only affects algorithms which require ground state suggestions.

Creation and Optimization of Ansatz

The selection of ansatz is a critical step in the VQE. There are two main approaches to choosing an ansatz: chemically motivated ansatz, such as the unitary coupled cluster method, or "device ansatz" which are easiest to implement given experimental constraints [1]. For our project, we implemented the latter. Specifically, we give the user the option to run a naïve variational eigensolver which uses the ground state provided in Broombridge as the ansatz. Alternatively, the user can select a true variational eigensolver which uses the creation/annihilation operator information from Broombridge and the Nelder-Mead (NM) simplex based direct search method to identify the optimal parameters which yield the ideal ground state.

To investigate the robustness of our VQE, we ran perturbation experiments on the initial conditions for the NM method. We used the ground state data from Broombridge and decreased the coeffecients of the creation/annihilation operators by 5, 10 and 20 percent to investigate the effect on the final solution found by the NM method. We conducted the perturbation experiment on the water molecule, H_2O. The results are summarized in the table below:

	0%	5%	10%	20%
μ	1.000	0.950	0.900	0.800
σ	1.22e-5	1.12e-5	5.11e-6	8.88e-6

The results from the perturbation experiments suggest that there is a strong coupling between the initial condition vector passed to the NM

algorithm and the final result. We believe this is due to the stochastic nature of the problem and the inherent randomness of the expectation value code. In future iterations, we believe that our implementation of the VQE would benefit from a technique such as stochastic gradient descent rather than NM to eliminate the coupling between initial conditions and the result of the optimization.

Impact

Together, our work spans the direct simulation of Hamiltonians to the integration of the Microsoft Quantum Chemistry Library with PSI4. Our research accelerates the capabilities of near-term / NISQ devices and enables quantum devices running Q# to immediately tackle small to medium scale Hamiltonian simulations. Because of the VQE, the simulations run are much shallower with comparable performance to existing methods. In addition, the PSI4 to Broombridge conversion utility extends the Quantum Chemistry library to other computational chemistry packages. This allows the VQE code developed, and other simulative methods like Trotterization and Qubitization, to be run with the PSI4 format.

Conclusion

In conclusion, our work formalizes an end-to-end system for NWChem and PSI4 files to run VQEs on emerging NISQ devices. We demonstrate that implemented models have accuracy comparable to classical polynomial time algorithms (SCF) and have minimal variability during simulation. This research empowers future devices running Q# code to tackle problems in quantum chemistry immediately.

VQEs do have limitations in precision, however, as the sampling technique used requires exponential numbers of samples to achieve specific levels of confidence. In addition, the overhead in producing the initial ansatz state is costly in comparison to Trotterization, which requires fewer preparations. Thus, there is a trade-off between coherence times, gate availability, and the simulation method of choice.

Future Work

For future work, smart parameter update schemes during ansatz preparation could enable a reduction in the number of prepared ansatzes, allowing for leaner runtimes when identifying ground states. There is interest in leveraging machine learning to identify smarter update schemes for the parametrized ansatz, as opposed to simply using Nelder-Mead.

Additional work could also focus on adding additional conversion systems to Broombridge. By connecting more compuational chemistry packages to Microsoft's Quantum Chemistry Libraries, Q# could simulate more molecules with greater precision.

There is also interest in utilizing alternative transformations. The Jordan-Wigner transformation is used because of its relative simplicity - however, there are other transformations (like the Bravyi-Kitaev transformation) which may be preferable [11]. Integrating the VQE and Q# library with these transformations would enable greater control when simulating molecules.

Acknowledgements

We thank Dr. Krysta Svore and Dr. Guang Hao Low for their support on the quantum chemistry aspects of the project. Additionally, we thank Mariia Mykhailova for her advice on the Q# components of the project.

This work was made possible through the Quantum Development Kit provided by Microsoft, the NWChem package provided by Pacific Northwest National Laboratory, and the Paul G. Allen School for Computer Science & Engineering at the University of Washington.

References

[1] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

[2] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences*, 114(29):7555–7560, 2017.

[3] Dave Wecker, Matthew B Hastings, Nathan Wiebe, Bryan K Clark, Chetan Nayak, and Matthias Troyer. Solving strongly correlated electron models on a quantum computer. *Physical Review A*, 92(6):062318, 2015.

[4] Jonathan Olson, Yudong Cao, Jonathan Romero, Peter Johnson, Pierre-Luc Dallaire-Demers, Nicolas Sawaya, Prineha Narang, Ian Kivlichan, Michael Wasielewski, and Alán Aspuru-Guzik. Quantum information and

computation for chemistry. *arXiv preprint arXiv:1706.05413*, 2017.

[5] James Daniel Whitfield, Peter John Love, and Alan Aspuru-Guzik. Computational complexity in electronic structure. *Physical Chemistry Chemical Physics*, 15(2):397–411, 2013.

[6] nathanwiebe2. Simulating hamiltonian dynamics, Oct 2017.

[7] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[8] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[9] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.

[10] et al Bylaska, EJ, WA De Jong, N Govind, K Kowalski, TP Straatsma, M Valiev, D Wang, E Apra, TL Windus, J Hammond, et al. Nwchem, a computational chemistry package for parallel computers, version 5.1. *Pacific Northwest National Laboratory, Richland, Washington*, 99352:0999, 2007.

[11] Jacob T Seeley, Martin J Richard, and Peter J Love. The bravyi-kitaev transformation for quantum computation of electronic structure. *The Journal of chemical physics*, 137(22):224109, 2012.

A Algorithm Pseudocode

Algorithm 1 Expectation Value Finding

term ← a single Jordan Wigner transformation term
qubitarray ← ancilla
errormargin ← 1
while *errormargin* > *threshold* **do**
    Prepare state on *qubitarray*.
    Apply all terms in *term*
    Measure eigenvalues on the *term* basis
    Record the eigenvalue found
    Update the predicted expectation value
    errormargin ← confidence interval spread

Algorithm 2 VQE

$\theta \leftarrow$ ansatz state parameters
pauliterms ← Separate Jordan Wigner transformation terms
runsum ← 0
for *term* in *pauliterms* **do**
    $runsum \leftarrow runsum + Algorithm1(term)$
if $runsum = \min_\theta runsum$ **then**
    Ground state/energy found $(\theta, runsum)$
else
    $\theta \leftarrow \theta'$ by optimization
    goto *top*.