TECHNICAL REPORT 3330
JANUARY 2024

Analysis of Heisenberg Models Using Variational Quantum Eigensolvers

Sean Crowe
Ramiro Rodriguez
NIWC Pacific

John Stenger
Naval Research Laboratory

Jaecob Adajar
Jarod Afalla
Southwestern College

Naval Information Warfare Center (NIWC) Pacific
San Diego, CA 92152-5001

This page is intentionally blank.

TECHNICAL REPORT 3330
JANUARY 2024

Analysis of Heisenberg Models Using Variational Quantum Eigensolvers

Sean Crowe
Ramiro Rodriguez
NIWC Pacific

John Stenger
Naval Research Laboratory

Jaecob Adajar
Jarod Afalla
Southwestern College

Administrative Notes:
This report was approved through the Release of Scientific and Technical Information (RSTI) process and formally published in the Defense Technical Information Center (DTIC) in January 2024.

NIWC Pacific
San Diego, CA 92152-5001

NIWC Pacific
San Diego, California 92152-5001

P. M. McKenna, CAPT, USN
Commanding Officer

M. J. McMillan
Executive Director (Acting)

ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Cryogenics Electronics and Quantum Research Branch _Branch (Code 71730) of the Basic and Applied Research Division, Naval Information Warfare Center Pacific (NIWC Pacific), San Diego, CA. The NIWC Pacific Naval Innovative Science and Engineering (NISE) Program provided funding for this Basic Applied Research project..

Released by
John deGrassie, Division Head
Basic and Applied Research Division

Under authority of
Carly Jackson, Department Head
Cyber / Science & Technology Department

ACKNOWLEDGMENTS

This work was supported under the NIWC-PAC In-House Innovation Program.

Editor: LMG

EXECUTIVE SUMMARY

Objective

The analysis of spin systems[1] is a natural use case for quantum computers because quantum bits (qubits) are generally created to emulate the behavior of quantum spins. Classical computers are not able to efficiently capture general features of spin systems because their state space grows exponentially with the number of particles. This work aims to use a quantum computer to analyze the ground-state energy of an anti-ferromagnetic Heisenberg model on a Kagome lattice unit cell. Our approach to this problem is to use a variational quantum eigensolver.

Results

In this report we document the details of an analysis we performed on several Heisenberg models with different geometries. Our calculations used quantum simulators and real quantum computers. We successfully prepared the ground-state energy of a Heisenberg model on the IBM Quantum (IBM Q) Guadalupe machine and submitted our solution for consideration in the IBM Q Open Science Competition 2022 [2] [3].

Recommendations

Analysis of spin systems using quantum computers is an emerging research direction that is beginning to see significant progress due to breakthroughs in superconducting hardware and error mitigation techniques. We recommend that Naval Information Warfare Center (NIWC) Pacific continue investigating in this area of research.

[1] Spin systems encompass a broad range of models, including spinning rigid bodies, Ising models, and Heisenberg models.

[2] More information about the IBM Quantum Open Science Competition 2022 is available at: `https://research.ibm.com/blog/ibm-quantum-open-science-prize-2022`

[3] For completeness we have included our full submission to the IBM Open Science Competition as a pdf in appendix A.

CONTENTS

Figures

1. Introduction

The simulation of quantum systems on classical computers is generally difficult owing to the exponential growth of the Hilbert space[4] with the number of particles in the quantum system. Studies of many-body quantum systems are therefore currently limited to a few solvable models, toy models with a small problem size, or perturbations around solvable models. This limitation is general and shows in up in a broad spectrum of fields ranging from quantum chemistry to quantum gravity [1–3]

A major potential application of quantum computers is the simulation of physical systems, which cannot be done on a classical computer [4]. This type of emulation is possible because the quantum computer can naturally support the entanglement structure of the quantum system being analyzed. Notably, noisy intermediate scale quantum computers were recently used to analyze a 127-site Ising model [5] in regimes where tensor network techniques and brute force analysis both break down. This analysis was made possible by recent breakthroughs in error mitigation techniques [6, 7].

Spin systems are a a natural starting point to explore applications of quantum computers to the analysis of quantum systems. This is because individual quantum mechanical spin $1/2$ particles only have two states: spin up and spin down. These spin states can be naturally mapped onto the states of a qubit: $|0\rangle$, and $|1\rangle$. An interesting and simple model of interactions between spins is the Heisenberg model. This model is characterized by the following Hamiltonian: $H = -\sum_{\langle i,j\rangle} J_{ij}\vec{\sigma}_i \cdot \vec{\sigma}_j + \sum_i \vec{B}_i \cdot \vec{\sigma}_i$. Where $\langle ij\rangle$ means to sum over nearest neighbor pairs, J_{ij} is a matrix that encodes interactions between different neighbors, and $\vec{B}_i$ is the magnetic field acting at the i^{th} site. Additionally, $\vec{\sigma}_i$ is a vector of Pauli matrices at the i^{th} lattice site. Specifically, $\vec{\sigma} = \sigma_x\hat{i} + \sigma_y\hat{j} + \sigma_z\hat{k}$, and

$$\begin{aligned} \sigma_x &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \sigma_y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\ \sigma_z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \end{aligned} \tag{1}$$

The main contribution of this document is to present a method for preparing the ground-state energy of a Heisenberg model on a Kagome lattice unit cell (see Fig. 3) using the IBM Q Guadalupe quantum computer. This work was performed for the IBM Q Open Science Competition 2022. The competition constrained that we were required to use variational quantum eigensolving to approach this problem, but otherwise we were free to use any qiskit packages, or python packages as long as they are free.

This document is organized as follows. Section two details Heisenberg models on smaller lattices that can be solved analytically. These analytical models can give insight into the solutions we eventually develop for the full Kagome unit cell. Section three discusses the construction of a quantum circuit ansatz, which can prepare the Kagome lattice cell ground state on a classical computer that simulates the quantum dynamics. This step is important because it allows us to debug our circuit before proceeding to execute on the real quantum computer, which has long queue times. Additionally, in section three we discuss the selection of the optimizer we use to vary the parameters of our quantum circuit as well as the initial conditions we chose for our variational circuit. In section four, we discuss details related to how we executed our algorithm on real quantum hardware. In particular, details related to error mitigation, and system truncation through symmetry are discussed. Finally in section five, we discuss extensions of our method to Heisenberg models on a single graphene molecule.

[4] The Hilbert space is a vector space equipped with an inner product, and its vectors describe the state of the system.

2. Exact Ground State(s)

2.1 Simple models

Before moving onto a numerical analysis of the full Kagome unit cell, we first consider here the analytic diagonalization of the two- and three-site Heisenberg models. These simpler models offer insights that might be missed in the full model because of its technical complexity.

2.1.1 Two-site Heisenberg model

The simplest Heisenberg model we can consider that still captures features of the full unit cell is the Heisenberg model on a two-site graph. Similar to the potential energy of two coupled magnetic dipoles, this two-site model is described by the Hamiltonian,

$$\hat{H} = \vec{\sigma}_1 \cdot \vec{\sigma}_2. \tag{2}$$

Where $\vec{\sigma}$ is a vector containing the three Pauli matrices. For pure states, $\langle \vec{\sigma} \rangle$ is a unit vector. On a classical level, this Hamiltonian is minimized when the the spins at the two different sites are anti-aligned and the classical ground-state energy is $E_g^{\text{classical}} = -1$[5]. However, when quantum entanglement between the two lattice sites is accounted for, lower energies can be reached.

The basis of quantum states for this model is given by $\mathcal{B}_2 = \{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$. Expressed in this basis, the Hamiltonian is,

$$\hat{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

$|00\rangle$ and $|11\rangle$ are eigenstates of this Hamiltonian with eigenvalue$= +1$. Because these two eigenstates are not entangled, this value agrees with the classical case where both spins are aligned.

To find the last two energy eigenvalues of this system, we have to consider the subspace spanned by $\mathcal{S} = \{|10\rangle, |01\rangle\}$. The Hamiltonian restricted to this subspace is

$$\hat{H}_{\mathcal{S}} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}. \tag{4}$$

Using standard linear algebra, the eigenvectors of this Hamiltonian are found to be $|a\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$ and $|g\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle)$. The energy eigenvalues are $E_a = +1$ and $E_g = -3$. Putting all of this together, the eigenbasis of this two-site Heisenberg model is $\mathcal{B}_e = \{|00\rangle, |11\rangle, |a\rangle, |g\rangle\}$, with respective energy eigenvalues $E = \{1, 1, 1, -3\}$. The ground state is then $|g\rangle = \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle)$ with energy eigenvalue $E_g = -3$.

Using the language of quantum circuits, we can express the ground state using standard quantum gates. The ground state $|g\rangle$ can be prepared using the quantum circuit shown in Fig. 1. This representation of the ground state as a quantum circuit will be useful later on when we construct the ground state of the full model using a quantum circuit.

[5] Unless state otherwise, all energies in this report are dimensionless

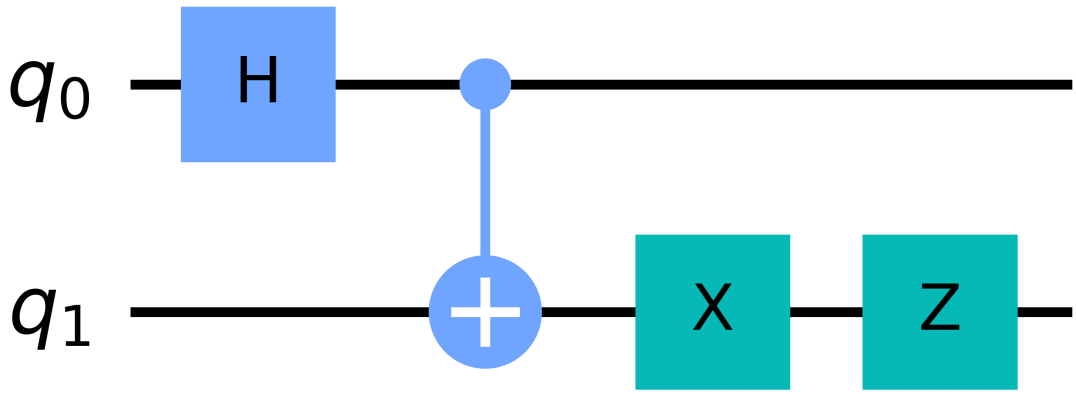


Figure 1. Two qubit circuit which prepares the ground state of the two-site Heisenberg model. The Hadamard gate and controlled-not (CNOT) gate create a Greenberger–Horne–Zeilinger state (GHZ state) and then X and Z Pauli matrices are applied to the second qubit.

2.1.2 Three-site Heisenberg model

We now move onto a three-site model corresponding to the graph shown in Fig. 2. This model now includes interactions between three sites on the lattice and therefore described by the following Hamiltonian,

$$\hat{H} = \vec{\sigma}_1 \cdot \vec{\sigma}_2 + \vec{\sigma}_2 \cdot \vec{\sigma}_3 + \vec{\sigma}_3 \cdot \vec{\sigma}_1. \tag{5}$$

At a classical level this system is more interesting than the previous one. It is not immediately clear what its classical ground-state energy should be because not all three spins can be anti-aligned at the same time. Proceeding to the quantum case, this system has the following set of states as a basis,

$$\mathcal{B}_3 = \{|000\rangle, |100\rangle, |010\rangle, |110\rangle, |001\rangle, |101\rangle, |011\rangle, |111\rangle\}\,. \tag{6}$$

Expressed in this basis, the Hamiltonian takes the following form,

$$\hat{H} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 2 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}. \tag{7}$$

Despite the larger size of this system, we can still find its spectrum analytically using computer algebra. The energy eigenvalues of this system are, $E = \{3, 3, 3, 3, -3, -3, -3, -3\}$. Additionally, the respective

eigenvectors for this system are,

$$
\begin{aligned}
|e1\rangle &= |000\rangle \\
|e2\rangle &= |111\rangle \\
|e3\rangle &= \frac{1}{\sqrt{3}}\left(|100\rangle + |010\rangle + |001\rangle\right) \\
|e4\rangle &= \frac{1}{\sqrt{3}}\left(|110\rangle + |101\rangle + |011\rangle\right) \\
|g1\rangle &= \frac{1}{\sqrt{2}}\left(|100\rangle - |010\rangle\right) \\
|g2\rangle &= \frac{1}{\sqrt{2}}\left(|100\rangle - |001\rangle\right) \\
|g3\rangle &= \frac{1}{\sqrt{2}}\left(|011\rangle - |101\rangle\right) \\
|g4\rangle &= \frac{1}{\sqrt{2}}\left(|011\rangle - |110\rangle\right).
\end{aligned}
\tag{8}
$$

The ground state is therefore quadruplely degenerate with energy eigenvalue $E_g = -3$.

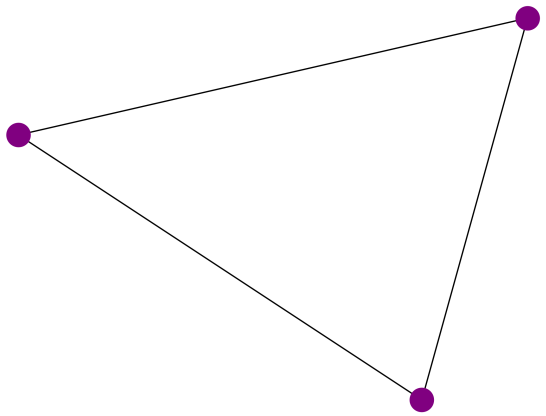


Figure 2. Plot of three-site graph on which we implement a Heisenberg model.

2.2 Numerical Analysis of Full System

We now move onto the diagonalization of the full Kagome unit cell. A graph of this unit cell is shown in Fig. 3. We implement a Heisenberg model by including an interaction term for each link on this graph by using the Hamiltonian,

$$
\begin{aligned}
\hat{H} = &\vec{\sigma}_1 \cdot \vec{\sigma}_2 + \vec{\sigma}_2 \cdot \vec{\sigma}_3 + \vec{\sigma}_3 \cdot \vec{\sigma}_1 \\
&+ \vec{\sigma}_3 \cdot \vec{\sigma}_4 + \vec{\sigma}_4 \cdot \vec{\sigma}_5 + \vec{\sigma}_5 \cdot \vec{\sigma}_3 \\
&+ \vec{\sigma}_5 \cdot \vec{\sigma}_6 + \vec{\sigma}_6 \cdot \vec{\sigma}_7 + \vec{\sigma}_7 \cdot \vec{\sigma}_5 \\
&+ \vec{\sigma}_7 \cdot \vec{\sigma}_8 + \vec{\sigma}_8 \cdot \vec{\sigma}_9 + \vec{\sigma}_9 \cdot \vec{\sigma}_7 \\
&+ \vec{\sigma}_9 \cdot \vec{\sigma}_{10} + \vec{\sigma}_{10} \cdot \vec{\sigma}_{11} + \vec{\sigma}_{11} \cdot \vec{\sigma}_9 \\
&+ \vec{\sigma}_{11} \cdot \vec{\sigma}_{12} + \vec{\sigma}_{12} \cdot \vec{\sigma}_1 + \vec{\sigma}_1 \cdot \vec{\sigma}_{11}
\end{aligned}
\tag{9}
$$

This Hamiltonian is considerably more complicated than the ones analyzed in 2.1. Because Pauli matrices at 12 different lattice sites are combined through tensor products, this Hamiltonian acts on a vector space with dimension $d = 2^{12} = 4096$. Analytic solutions for the ground state of this Hamiltonian are still available as we will see later on. However, writing these solutions explicitly as in 2.1 is not practical.

Numerically, we find the first few eigenvalues to be $E = \{-18, -18, -16.96, -16.96, -16.96, ...\}$. The ground state is therefore doubly degenerate with eigenvalue $E_g = -18$. We have found all energy eigenvalues using standard linear algebra routines from the Python-library NumPy[6]. The results are shown in histogram below in Fig. 4

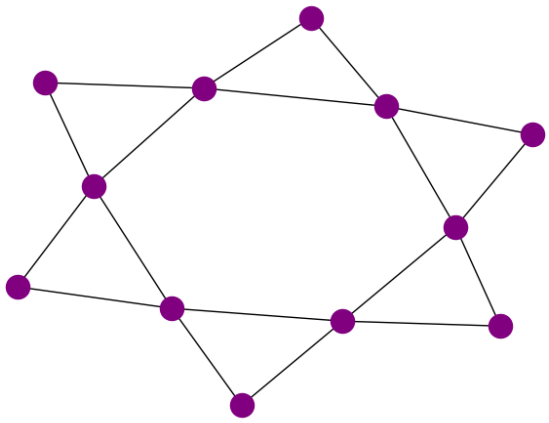


Figure 3. Graph of a Kagome lattice unit cell.

3. Simulated Variational Quantum Eigensolving (VQE)

In this section we discuss the details related to the construction of the ground state of a Heisenberg model on a Kagome unit cell on a classical computer. Kagome unit cells have 12 nodes, meaning the Hilbert space needed to describe them is still small enough that it can be simulated on a classical computer, but complex enough that implementation on a quantum computer is not trivial. In this way, first performing the analysis on a classical computer in preparation for a run on the real quantum computer can allow for faster prototyping of quantum circuits because long queueing times on real quantum computers can be circumvented.

3.1 Ansatz Selection and Training

Generally, the way variational quantum eigensolving works is that a quantum circuit with tunable parameters is chosen. This quantum circuit will prepare a quantum state $|\psi(\theta)\rangle$, which depends on the tunable parameters θ. From there, many shots are taken of the circuit in order to estimate the expectation value of the Hamiltonian, $\langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle$, where $\hat{H}$ is given by (10). After the Hamiltonian is estimated, one varies the parameters in order to minimize this expectation value. According to variational principle in quantum mechanics, it is not possible to produce a value that is below the ground-state energy in this manner. Operationally speaking, this process is similar to training a neural network where the cost function can only be evaluated on a quantum computer.

[6] `https://numpy.org/`

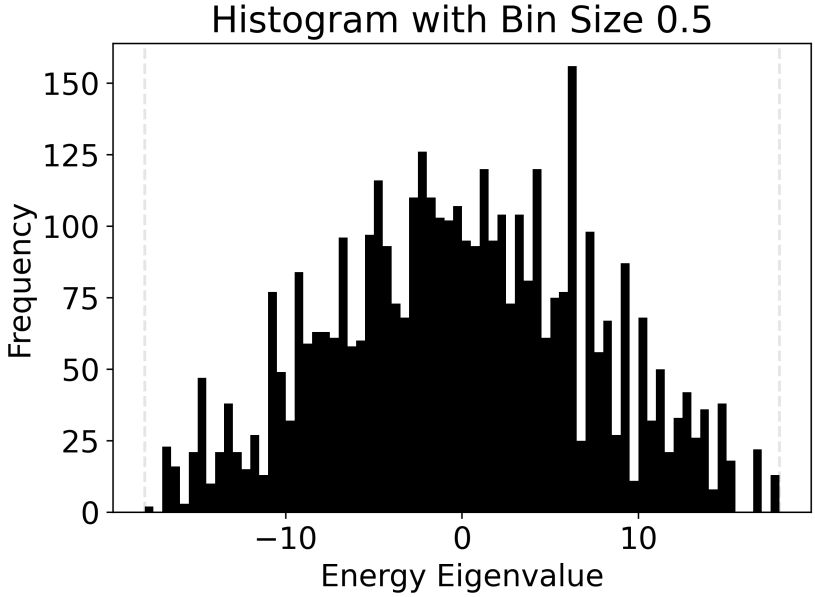


Figure 4. Histogram of allowed energy eigenvalues for the Heisenberg model on a Kagome lattice unit cell. Dashed lines have been added at $E = \pm 18$ to indicate the minimum and maximum allowed energies.

The most difficult part of this process is selecting an ansatz that can prepare the ground state of the system which we are interested in. The second hardest part is making this ansatz efficient enough that it can be executed on a real quantum computer that is subject to noise. Our approach to this process was iterative and involved trial and error to select circuits that were efficient and could prepare the ground-state energy.

An initial quantum circuit we selected is shown below in Fig. 5. This circuit has 143 tunable parameters and a depth of 13 and 17 CNOT gates, which are the most expensive to execute. However, this ansatz is able to prepare the ground-state energy. We trained this ansatz using the standard simultaneous perturbation stochastic approximation (SPSA) optimizer [8] with randomized initial conditions. The results for this training are shown below in Fig. 6. While this circuit is able to prepare the ground state after sufficient training, it takes too many iterations to be considerable for the competition. Moreover, the circuit is not shallow, and its depth could lead to unacceptable levels of noise buildup in the final quantum state. For these reasons this quantum circuit was not included in our final submission to the competition.

After much trial and error, we were able to reduce the previous circuit to the one shown in Fig. 7. This ansatz has a periodic symmetry and is notable because not all qubits are connected by CNOT gates, which implies that this system is not fully entangled, meaning its quantum state can be expressed analytically. It is not clear whether the organizers of the competition intended for such a solution to be available. Additionally, this ansatz has a depth of 6 gates and only 24 trainable parameters, which are included towards the end of the circuit in a layer of rotation gates. This circuit prepares the ground state when all parameters are set to zero. The rotation gates are therefore only included to compensate for systematic errors that may be present on the physical quantum computer.

We have trained this model using the standard SPSA optimizer included with Qiskit. SPSA is designed to handle noisy optimization landscapes, making it a good choice for optimizing quantum algorithms on noisy quantum computers. Our initial conditions were selected to be close to the analytically known

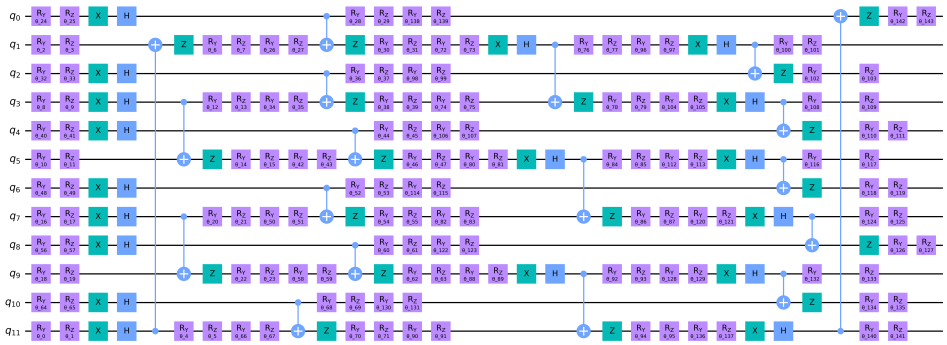


Figure 5. A quantum circuit that can prepare the ground-state energy of a Heisenberg model on a Kagome Lattice unit cell.

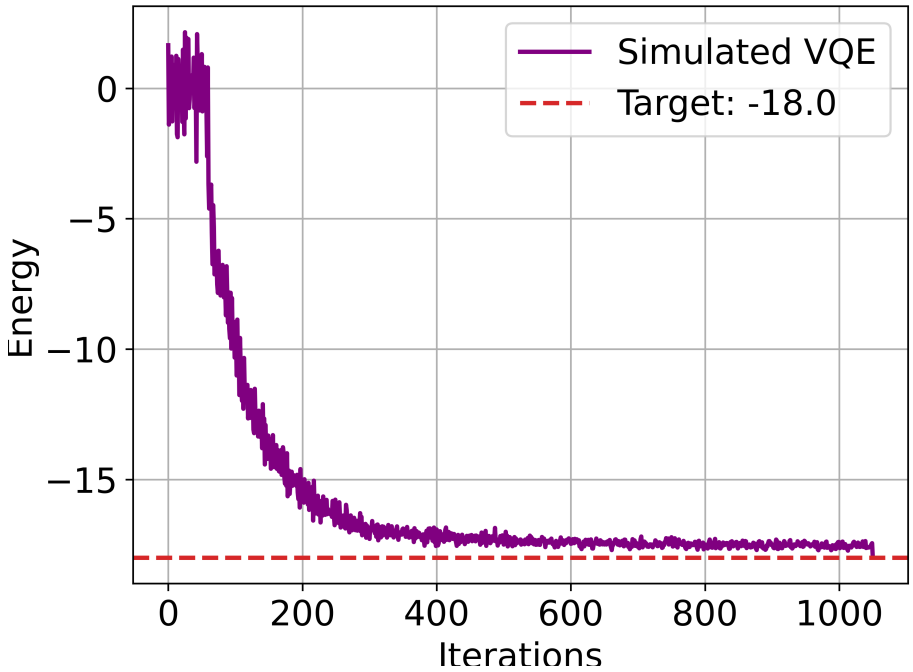


Figure 6. Training data from the above quantum circuit in Fig. 5. Optimizer is able to acheive sub-percentage error in just over a thousand iterations.

solution corresponding to the case where all parameters are set to zero. Starting from a slightly perturbed initial state can help compensate for systematic errors, which are small. Training data for this setup can be found below in Fig. 8. The optimizer is able to achieve percentage level closeness with the ground state after only a few iterations and it is able to achieve machine precision with the ground state after only 600 iterations. Of course, achieving machine level precision on a quantum computer will not be possible on the IBM Q Guadalupe machine because of noise considerations. However, it is encouraging to see this ansatz capable of preparing the ground state exactly using a low number of training iterations and circuit depth.

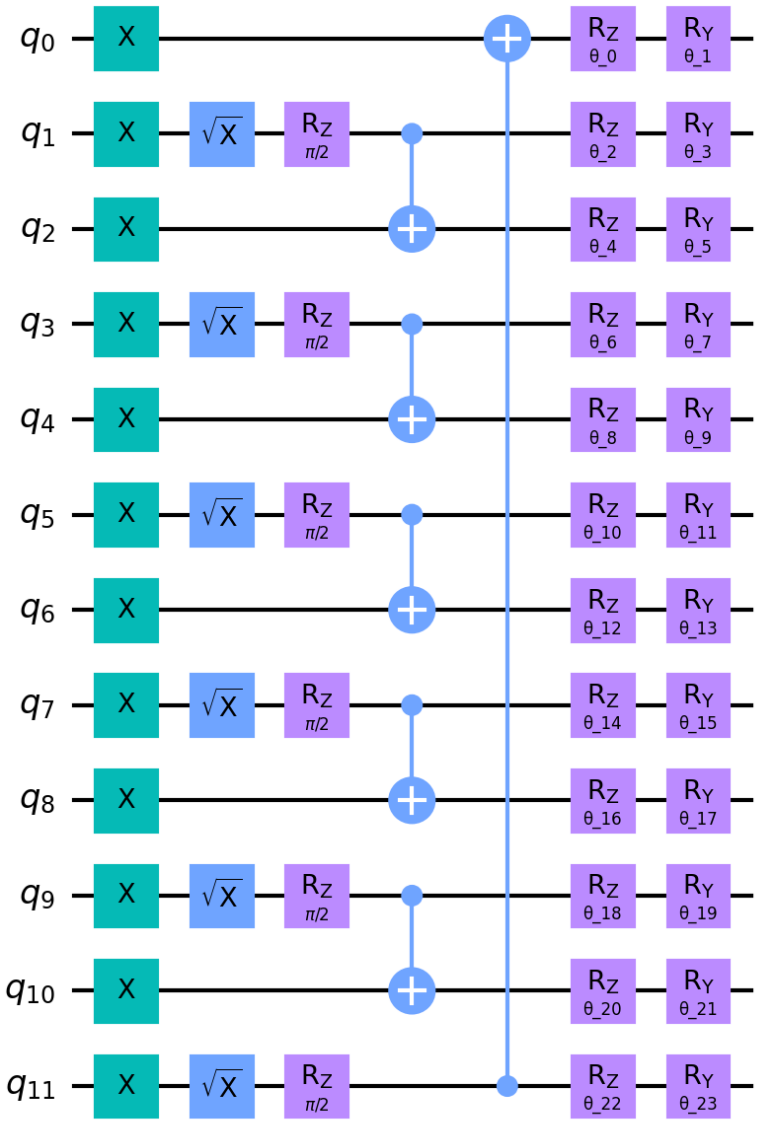


Figure 7. Quantum circuit ansatz that is able to prepare the ground-state energy with a very short depth.

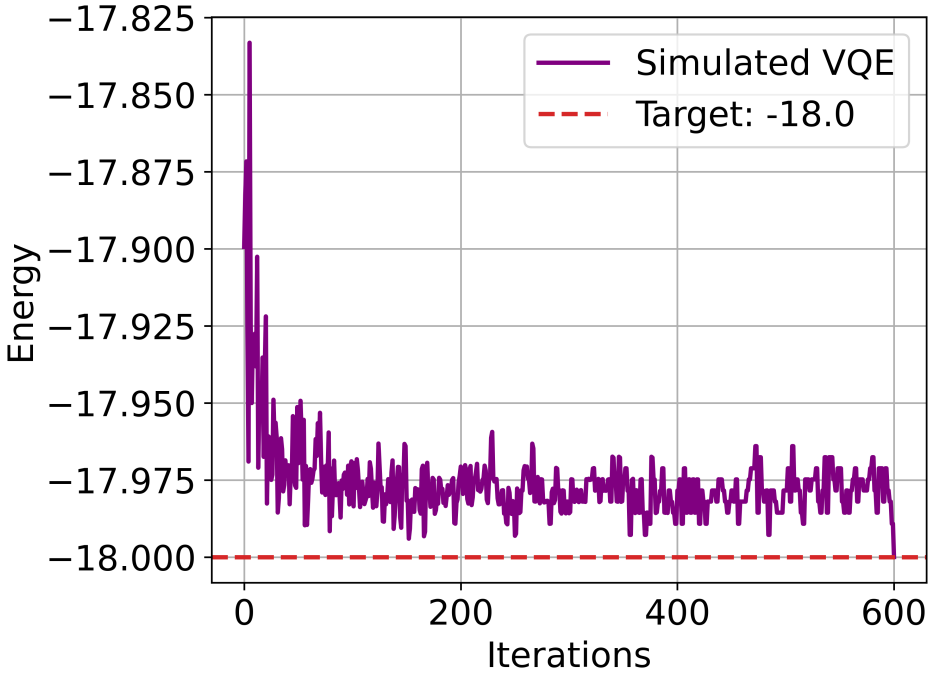


Figure 8. Training data for the ansatz shown in Fig. 7.

4. Physical VQE

Having discussed the construction of our circuit ansatz, and training considerations such as initial conditions and optimizer selection, we can now move on to discussing details related to executing our algorithm on a real quantum computer. As per the IBM Open Science Competition rules, we are constrained to execute our algorithm on the IBM Q Guadalupe machine. Guadalupe is a 16 qubit quantum computer with a topology similar to the Kagome unit cell. It has median coherence times of $T_1 = 82\,\mu$s and $T_2 = 103\,\mu$s. It has a median readout assignment error of 1.8% and a median CNOT error of 8.8×10^{-3}. Percentage level error bounds on the readout assignment make achieving an error of less than 1%[7] away from the ground state a non-trivial task.

An interesting feature of this ansatz is shown in Fig. 7 is that not all qubits are connected. There is one CNOT gate per triangle of the Kagome unit cell. This is a fact we exploit by focusing our solution on just one triangle of the Kagome unit cell. We then minimize the energy on just one of the triangles, and multiply by six in order to get the full ground-state energy. This approach has the advantage that fewer qubits are actually used and accumulation of readout errors is mitigated. Moreover, in this reduced ansatz there is only a single CNOT gate that can mitigate errors due to CNOT gates.

The reduced ansatz that we implement on Guadalupe is shown in Fig. 9. A final change we have made is to use $2 \times 10^{-3} \arctan(\theta)$ in the phase gates. This artificially restricts the parameters to a small range, which improves convergence.

Since we are focusing on one triangle of the unit cell, we modify the Hamiltonian to only include edges on that triangle. This allows for more accurate estimation of the energy because fewer terms in the

[7] According to the IBM Open Science Competition rules, solutions that do not achieve better than 1% accuracy will not be considered.

Hamiltonian need to be estimated. This truncated system is implemented by reducing the edge list to the links on one of the triangles before constructing the Hamiltonian using the Heisenberg model subroutine. The edge list we used was

```
edge_list = [
    (11, 14, t),
    (14, 13, t),
    (11, 13, t)
].
```

This corresponds to the triangle on the top of the unit cell. A figure showing this truncated unit cell can be seen in Fig. 11. After truncating the edge list, the Hamiltonian was constructed normally using the following code,

```
# Build Hamiltonian from graph edges
heis_16 = HeisenbergModel.uniform_parameters(
    lattice=kagome_unit_cell_16,
    uniform_interaction=t,
    uniform_onsite_potential=0.0,
)

# Map from SpinOp to qubits just as before.
log_mapper = LogarithmicMapper()
ham_16 = 6*4 * log_mapper.map(heis_16.second_q_ops().simplify()).
```

A factor of six was included in ham_16 because there are six triangles. To complete our ansatz, we transpiled it using optimization level three. The resulting transpiled circuit is shown in Fig. 10. It has a depth of 9 gates and only uses the three qubits corresponding to the upper triangle of the Kagome unit cell.

For our optimizer, we used the Qiskit SPSA function with custom parameters. Our optimizer was specified with the following command,

```
from qiskit.algorithms.optimizers import SPSA
optimizer = SPSA(maxiter=200,perturbation=0.05,learning_rate=0.00002).
```

The perturbation was selected to be similar in magnitude to the readout errors of the qubits we are using. This minimizes the relative error associated with our estimate of the gradient of the cost function. A relatively small learning rate was chosen because the optimal variational parameters are known to be zero for our circuit. We start our variational parameters at this analytically known solution, and a small learning rate helps to keep our solution from going too far away from this optimal point.

Evaluating circuits on real hardware generally needs special care. The jobs we send to Guadalupe use the following options,

```
options.execution.shots=99999
options.resilience_level = 1
options.optimization_level = 1.
```

We chose to max out the allowed number of shots. Additionally, our choice of resilience level and optimization provide a good balance between speed and accuracy.

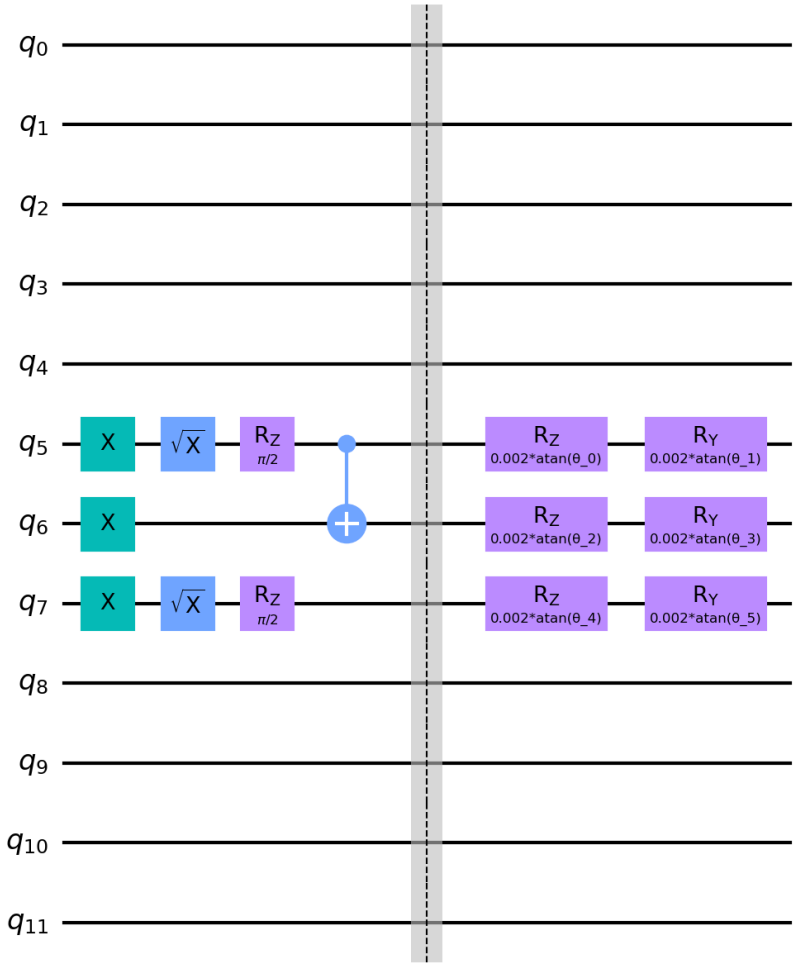


Figure 9. Truncated version of earlier ansatz shown in Fig. 7.

.

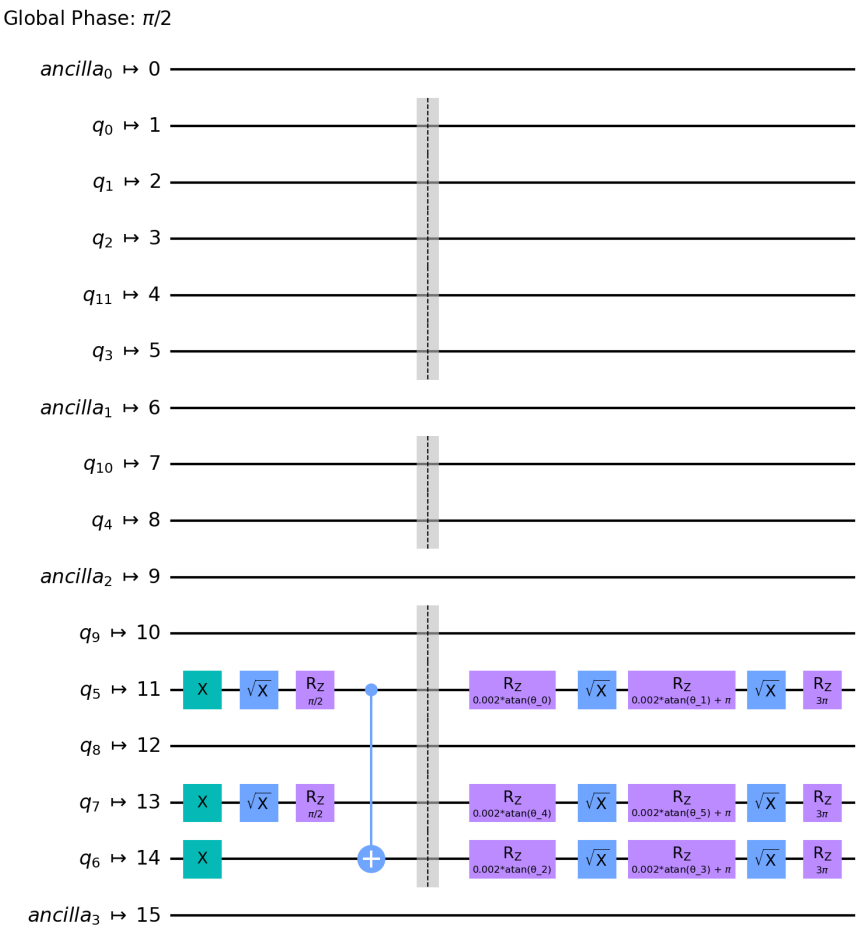


Figure 10. Transpiled version of our previous truncated circuit. The only change is that the Ry rotation gates are expressed as R_z gates and $\sqrt{x}$ gates, which can be natively implemented on Guadalupe. Additionally, qubit numbers have been changed to reflect their physical qubit numbers on the Guadalupe machine.

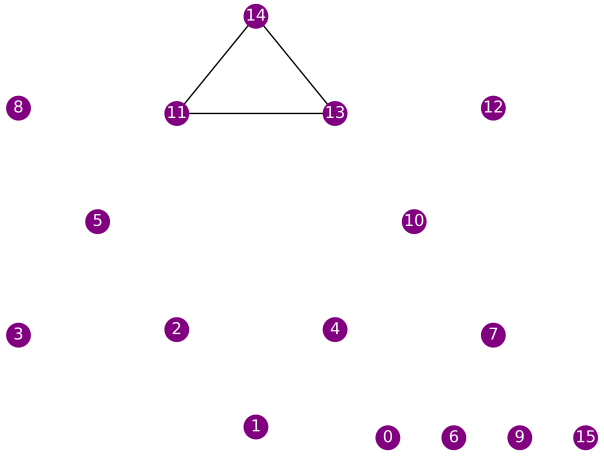


Figure 11. Truncated unit cell where only interactions necessary for preparing the ground-state energy are retained.

In addition to mitigating errors on the physical hardware, classically controlling our algorithm is also challenging. This is because we must access Guadalupe remotely, and queuing times can be between days and months depending on how busy the quantum computer is. If our connection to Guadalupe goes down even momentarily, we lose our place in the queue and must start over again. To mitigate this, we have executed our algorithm from the High-Performance-Computing Center (HPCC) at NIWC Pacific. This was done by securely copying our codes to the Heisenberg machine there over a secure shell (SSH) connection and executing our algorithm using the NIWC Pacific local area network (LAN) connection to the internet. This was the most stable way we found of executing our algorithm while maintaining contact with the internet for an extended period of time.

Using the above ansatz, Hamiltonian, optimizer, and evaluation options, we have performed VQE on real quantum hardware for the Kagome system and the results of our training can be seen in Fig. 12. To mitigate noise, at each step, we take the mean of all previously obtained energies. Our estimate of the energy has a clear trend toward the ground-state energy. For the particular run reported in Fig. 12 the relative difference between the ground state and our prediction is $\approx 3 \times 10^{-3}$, putting us within the limit of consideration.

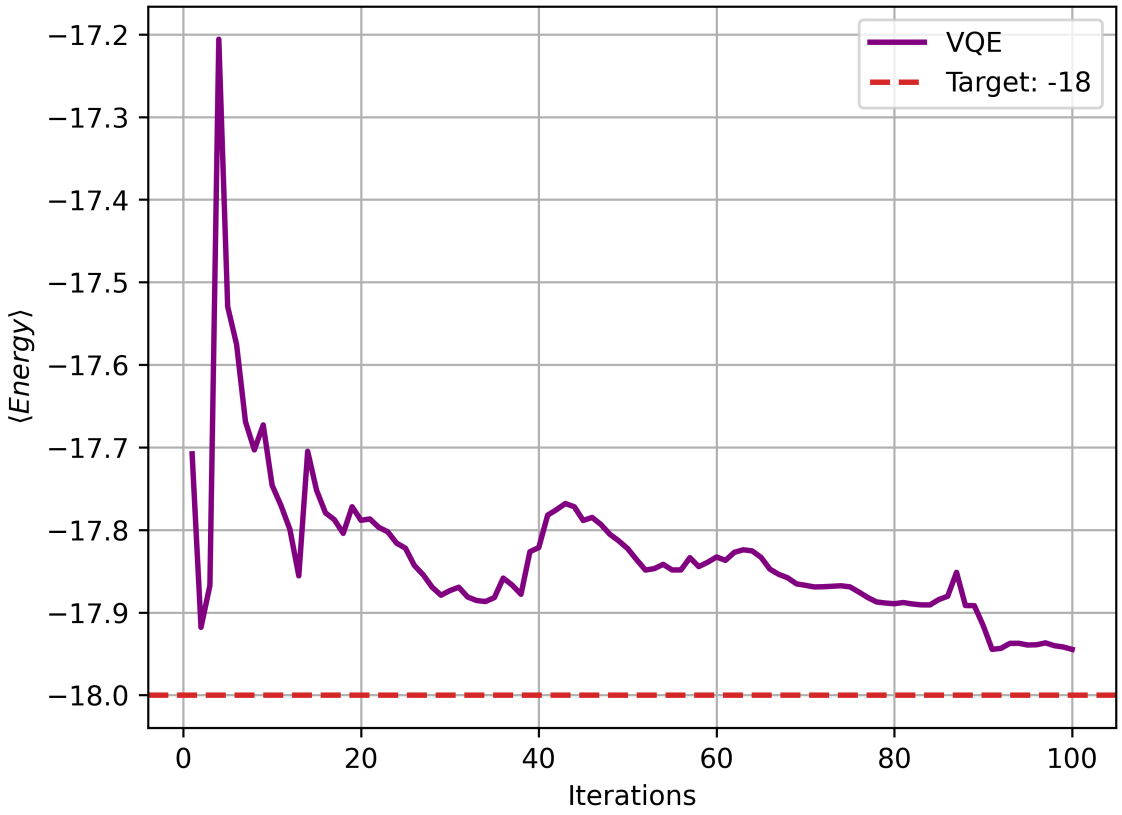


Figure 12. Training data for real execution on the IBM Q Guadalupe. Percentage level error rates were achieved after only 100 iterations.

5. Extensions to Graphene

In addition to exploring variational quantum algorithms for the Kagome lattice, we have also applied this technique to the case of a Heisenberg model on a single graphene unit cell. Heisenberg models on spin chains have been explored before and analytic solutions can be prepared using the Bethe Ansatz [9–12]. However, all of these known approaches use ancillary registers, which can complicate things by leading to more noise getting into the calculations. Our approach uses only a system register, which requires the same number of qubits as there are nodes in the unit cell. This graphene unit cell is shown below in Fig. 13, The nodes of the unit cell have been labelled by the qubit that will represent them on the Guadalupe machine. Because the graphene unit cell has only six nodes of interest, and because the Guadalupe machine has 16 available qubits, 10 of the qubits are simply along for the ride in this set of calculations. The Hamiltonian that describes this graphene unit cell is given by:

$$\begin{aligned}\hat{H} = \vec{\sigma}_1 \cdot \vec{\sigma}_2 + \vec{\sigma}_2 \cdot \vec{\sigma}_3 + \vec{\sigma}_3 \cdot \vec{\sigma}_4 \\ + \vec{\sigma}_4 \cdot \vec{\sigma}_5 + \vec{\sigma}_5 \cdot \vec{\sigma}_6 + \vec{\sigma}_6 \cdot \vec{\sigma}_1.\end{aligned} \tag{10}$$

This Hamiltonian acts on a vector space with $2^6 = 64$ dimensions. Classically, we expect this system to have a minimum energy of $E_g^{\text{classical}} = -6$ because all spins can be simultaneously anti-aligned. Quantum mechanically, this system has a ground state which below this value, and the ground-state energy eigenvalue was found using computer algebra to be $E_g = -2\left(2 + \sqrt{13}\right) \approx -11.21$. Moreover, this ground state is non-degenerate.

Initially, we approached this problem from a classical perspective by only including single qubit rotation gates. This approach is essentially classical because it is not possible to create entanglement between qubits by using single qubit rotation gates. This classically motivated circuit is shown in the Fig. 14. As expected, this circuit was able to reach classical ground state which can be seen in the training data shown in Fig. 15.

After completing an analysis at the classical level, we moved onto circuits with two qubit gates capable of creating entanglement. Several circuits were created and trained. A representative example is shown in Fig. 16. This circuit is essentially a combination of the classical circuit discussed above and the two-site circuit discussed earlier (Fig. 1). Despite having newer features, such as two qubit gates this circuit is not able to achieve energies lower than the classical value. It turns out to be difficult to develop a quantum circuit without additional registers that can be used to get below this classically allowed register. All told, 21 quantum circuits were analyzed and none of them were able to achieve an energy below -6.

This example points towards the limitations of the method of variational quantum eigensolving. Despite the generality of quantum circuits studied for this report, the ground-state energy was not able to be reached. Generally, some geometries and models will require special care and effort in order to complete an analysis of their ground state wavefunctions.

6. Conclusion

In this report we have reported the details of an analysis of a Heisenberg model on a Kagome lattice unit cell. Our approach employed a variational eigensolving to analyze the ground state of this system. Our quantum circuits were constructed using a trial and error approach, but were motivated by semiclassical and symmetrical considerations. Through the implementation of error mitigation techniques, we successfully attained levels of sub-percentage error rates necessary for consideration in the competition. Furthermore, our strategy to take advantage of the analytical properties of the solution proved useful, as we significantly reduced errors by truncating our system to a single triangle of the Kagome unit cell. Overall,

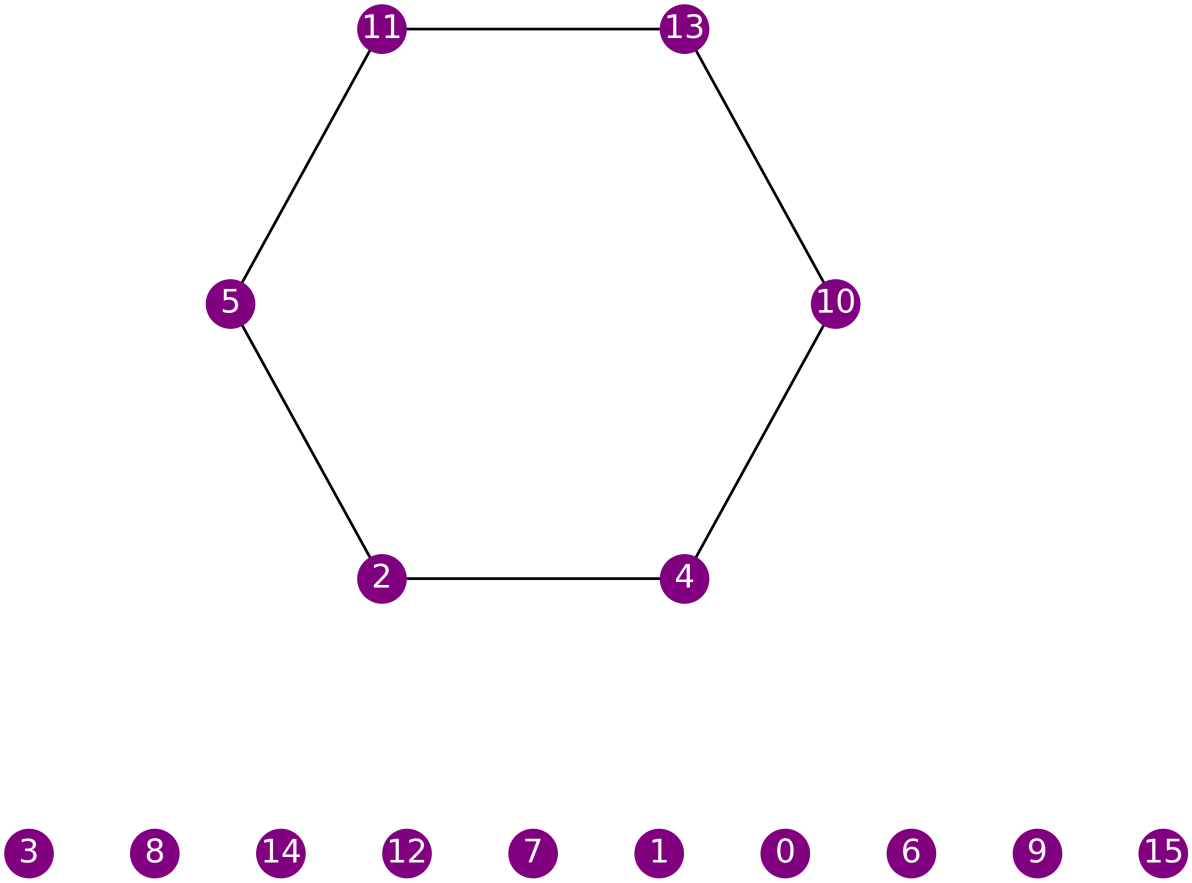


Figure 13. Graphene unit cell.

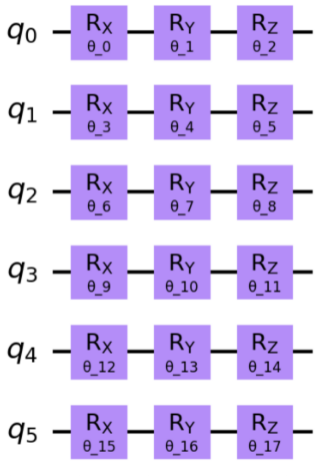


Figure 14. Variational quantum circuit for the graphene unit cell using only single qubit rotation gates.

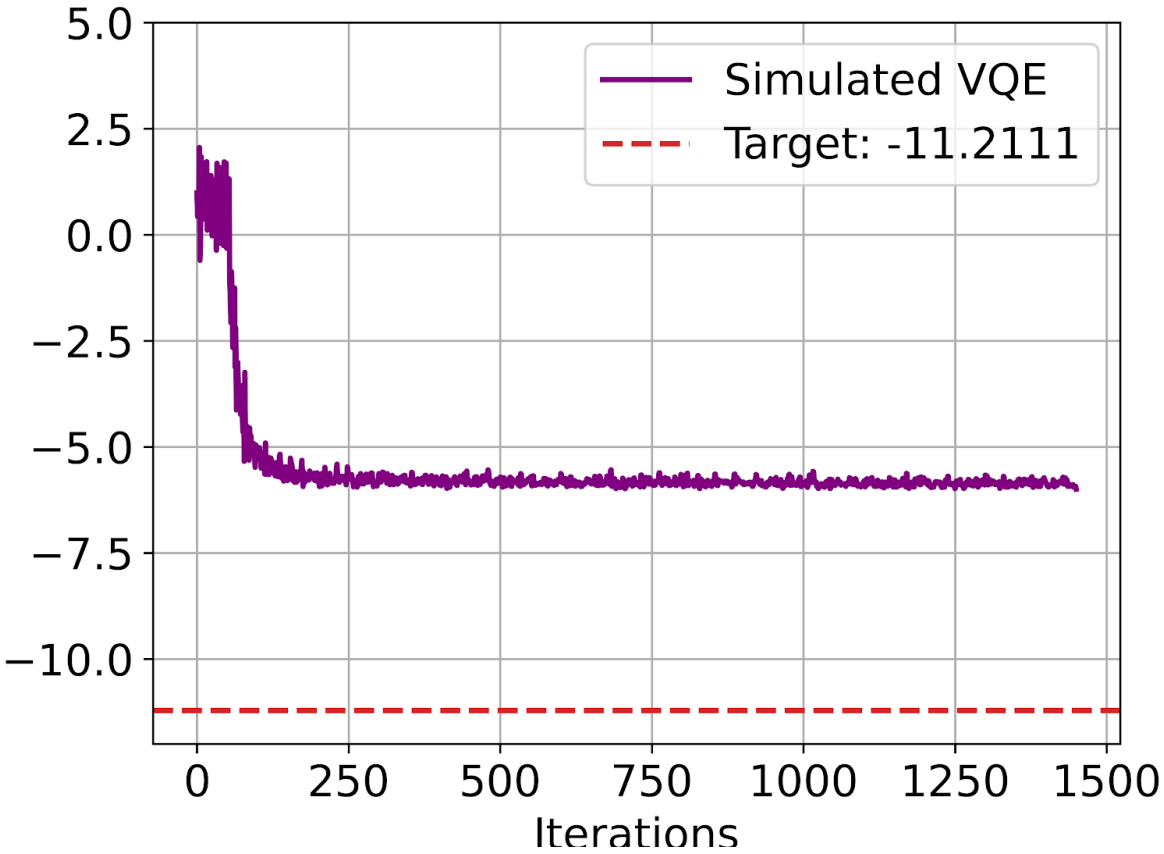


Figure 15. Training data for the classical circuit shown in Fig. 14.

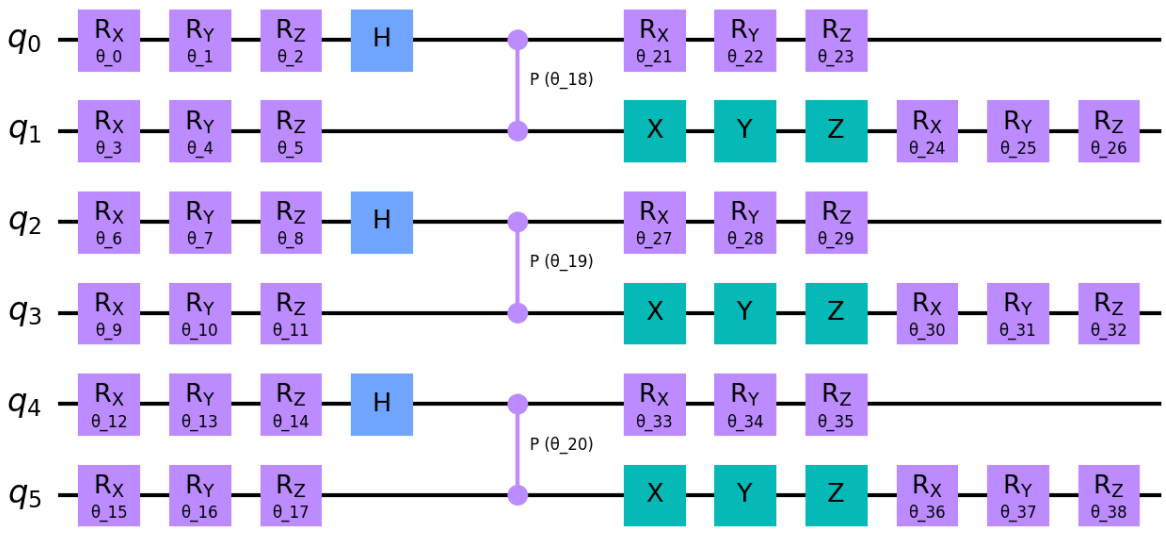


Figure 16. Quantum circuit for analyzing ground-state energy of a single graphene unit cell. The circuit begins and ends with a layer of tunable rotation gates along with a layer of quantum gates in the middle, which were motivated by Fig. 1, which was able to minimize the energy of a two-site Heisenberg model.

this multi-faceted approach put us within range of consideration for the IBM Open Science Competition and advances the understanding and exploration of quantum systems within the Kagome lattice.

Additionally, we have used the same methods to analyze a Heisenberg model on a graphene unit cell. In this case, we started with a classical analysis of single qubit rotation gates and then gradually began introducing two-qubit gates to create entanglement between qubits. Unfortunately, in this case we were not able to prepare the ground state using our variational quantum eigensolving approach. In fact, we were not able to achieve energies below the classically allowed minimum energy. This example points to difficulties with the VQE approach in that it requires inspiration for constructing quantum circuits and sometimes requires additional computational overhead, such as additional registers that have been shown to be useful for preparing Bethe Ansatz, which is relevant to this particular case [9–11].

Taken as a whole, this report indicates the usefulness of variational quantum eigensolving on quantum computers for the purpose of analyzing quantum systems that may be difficult to analyze on a classical computer. Presently, this technique requires some care to yield useful results, but as quantum computers and quantum algorithms increase in quality, VQE can become a useful tool for analyzing quantum systems relevant to the U.S. NAVY.

REFERENCES

1. Ambjørn, J., Görlich, A., Jurkiewicz, J., and Loll, R. 2012. "Nonperturbative quantum gravity," *Physics Reports*, vol. 519, no. 4-5, pp. 127–210, URL `https://doi.org/10.1016%2Fj.physrep.2012.03.007`.

2. Jeong, W., Stoneburner, S. J., King, D., Li, R., Walker, A., Lindh, R., and Gagliardi, L. 2020. "Automation of Active Space Selection for Multireference Methods via Machine Learning on

Chemical Bond Dissociation," *Journal of Chemical Theory and Computation*, vol. 16, no. 4, pp. 2389–2399, URL https://doi.org/10.1021/acs.jctc.9b01297.

3. Lloyd, S. 1996. "Universal Quantum Simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078.

4. Nielsen, M. A. and Chuang, I. L. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press.

5. Kim, Y., Eddins, A., Anand, S., Wei, K. X., van den Berg, E., Rosenblatt, S., Nayfeh, H., Wu, Y., Zaletel, M., Temme, K., and Kandala, A. 2023. "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, URL https://doi.org/10.1038/s41586-023-06096-3.

6. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., Fowler, A., Gidney, C., Giustina, M., Graff, R., Guerin, K., Habegger, S., Harrigan, M. P., Hartmann, M. J., Ho, A., Hoffmann, M., Huang, T., Humble, T. S., Isakov, S. V., Jeffrey, E., Jiang, Z., Kafri, D., Kechedzhi, K., Kelly, J., Klimov, P. V., Knysh, S., Korotkov, A., Kostritsa, F., Landhuis, D., Lindmark, M., Lucero, E., Lyakh, D., Mandrà, S., McClean, J. R., McEwen, M., Megrant, A., Mi, X., Michielsen, K., Mohseni, M., Mutus, J., Naaman, O., Neeley, M., Neill, C., Niu, M. Y., Ostby, E., Petukhov, A., Platt, J. C., Quintana, C., Rieffel, E. G., Roushan, P., Rubin, N. C., Sank, D., Satzinger, K. J., Smelyanskiy, V., Sung, K. J., Trevithick, M. D., Vainsencher, A., Villalonga, B., White, T., Yao, Z. J., Yeh, P., Zalcman, A., Neven, H., and Martinis, J. M. 2019. "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, URL https://doi.org/10.1038/s41586-019-1666-5.

7. Temme, K., Bravyi, S., and Gambetta, J. M. 2017. "Error Mitigation for Short-Depth Quantum Circuits," *Phys. Rev. Lett.*, vol. 119, p. 180509, URL https://link.aps.org/doi/10.1103/PhysRevLett.119.180509.

8. "Johns Hopkins University Applied Physics Laboratory," https://www.jhuapl.edu/SPSA/, accessed: [07/26].

9. Li, W., Okyay, M., and Nepomechie, R. I. 2022. "Bethe states on a quantum computer: success probability and correlation functions," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 35, p. 355305, URL https://doi.org/10.1088%2F1751-8121%2Fac8255.

10. Dyke, J. S. V., Barnes, E., Economou, S. E., and Nepomechie, R. I. 2022. "Preparing exact eigenstates of the open XXZ chain on a quantum computer," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 5, p. 055301, URL https://doi.org/10.1088%2F1751-8121%2Fac4640.

11. Dyke, J. S. V., Barron, G. S., Mayhall, N. J., Barnes, E., and Economou, S. E. 2021. "Preparing Bethe Ansatz Eigenstates on a Quantum Computer," *PRX Quantum*, vol. 2, no. 4, URL https://doi.org/10.1103%2Fprxquantum.2.040329.

12. Nepomechie, R. I. 2021. "Bethe ansatz on a quantum computer?" .

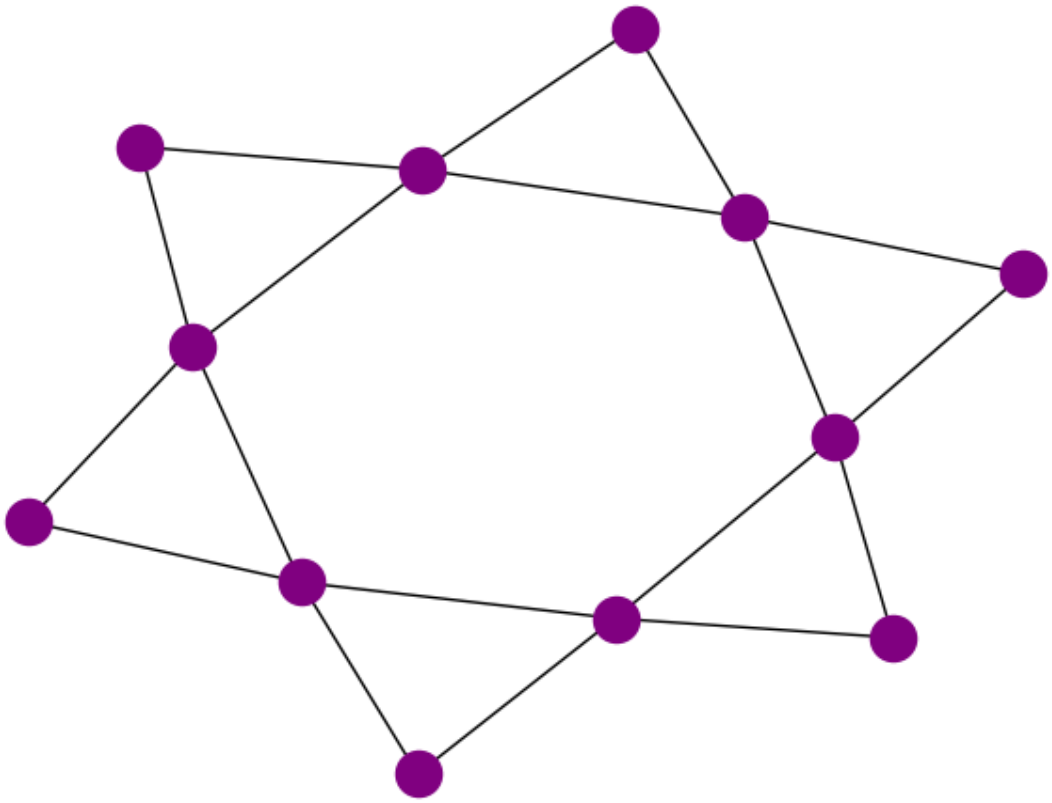
A. Python code submission

In [46]:
```
"""
Importing Packages
"""
import numpy as np

from time import time
import pickle
import matplotlib.pyplot as plt
plt.rcParams.update({"font.size": 16})  # enlarge matplotlib fonts
```

In [47]:
```
"""
Importing packages and appending to path
"""
import rustworkx as rx

from qiskit_nature.problems.second_quantization.lattice import Lattice
from matplotlib import pyplot as plt
# Custom Heisenberg couplings
import sys
sys.path.append(r'.')  # may be needed if running notebook on a cloud service
from heisenberg_model import HeisenbergModel
```

In [48]:
```
"""
Making an image of the Kagome Unit Cell
"""
# Kagome unit cell
num_sites = 12
# Edge weight
t = 1.0

# Generate graph of kagome unit cell
# Start by defining all the edges
graph = rx.PyGraph(multigraph=False)
graph.add_nodes_from(range(num_sites))
edge_list = [
    (0, 1, t),
    (1, 2, t),
    (2, 3, t),
    (3, 4, t),
    (4, 5, t),
    (5, 0, t),
    (0, 6, t),
    (1, 6, t),
    (1, 7, t),
    (2, 7, t),
    (2, 8, t),
    (3, 8, t),
    (3, 9, t),
    (4, 9, t),
    (4, 10, t),
    (5, 10, t),
    (5, 11, t),
    (0, 11, t),
]
# Generate graph from the list of edges
graph.add_edges_from(edge_list)
```

```
# Make a Lattice from graph
kagome_unit_cell = Lattice(graph)

# Draw Lattice
kagome_unit_cell.draw(style={'node_color':'purple'})
plt.savefig('kagome_unit_cell.png')
```



In [49]:
```
"""
IMPORT LOG MAPPER
"""
from qiskit_nature.mappers.second_quantization import LogarithmicMapper
```

In [50]:
```
"""
Build Hamiltonian from graph edges
"""
heis = HeisenbergModel.uniform_parameters(
    lattice=kagome_unit_cell,
    uniform_interaction=1.0,  # same spin-spin interaction weight as used in gr
    uniform_onsite_potential=0.0,  # No singe site external field
)

# The Lattice needs an explicit mapping to the qubit states.
# We map 1 qubit for 1 spin-1/2 particle using the LogarithmicMapper
log_mapper = LogarithmicMapper()
# Multiply by factor of 4 to account for (1/2)^2 terms from spin operators in t
ham = 4 * log_mapper.map(heis.second_q_ops().simplify())
# Print Hamiltonian to check it's what we expect.
# There are 18 edges and 3 terms per edge (XX, YY, and ZZ),
# so there should be 54 equally weighted terms.
ham
```

Out[50]:
```
PauliSumOp(SparsePauliOp(['ZIIIIIZIIIII', 'IZIIIIZIIIII', 'IZIIIIIZIIII', 'IIZ
IIIIZIIII', 'IIIIIIZZIIII', 'IIZIIIIIZIII', 'IIIZIIIIZIII', 'IIIIIIIIZZIII', 'I
IIZIIIIIZII', 'IIIIZIIIIZII', 'IIIIIIIIIZZII', 'IIIIZIIIIIZI', 'IIIIIZIIIIIZI',
'IIIIIIIIIZZI', 'ZIIIIIIIIIIZ', 'IIIIIZIIIIIZ', 'IIIIIIZIIIIZ', 'IIIIIIIIIIIZ
Z', 'YIIIIIYIIIII', 'IYIIIIYIIIII', 'IYIIIIIYIIII', 'IIYIIIIYIIII', 'IIIIIIYYI
III', 'IIYIIIIIYIII', 'IIIYIIIIYIII', 'IIIIIIIYYIII', 'IIIYIIIIIYII', 'IIIIYII
IIYII', 'IIIIIIIIYYII', 'IIIIYIIIIIYI', 'IIIIIYIIIIYI', 'IIIIIIIIIIYYI', 'YIIII
IIIIIIY', 'IIIIIYIIIIIY', 'IIIIIIYIIIIY', 'IIIIIIIIIIIYY', 'XIIIIIXIIIII', 'IXI
IIIXIIIII', 'IXIIIIIXIIII', 'IIXIIIIXIIII', 'IIIIIIXXIIII', 'IIXIIIIIXIII', 'I
IIXIIIIXIII', 'IIIIIIIXXIII', 'IIIXIIIIIXII', 'IIIIXIIIIXII', 'IIIIIIIIIXXII',
'IIIIXIIIIIXI', 'IIIIIXIIIIXI', 'IIIIIIIIIXXI', 'XIIIIIIIIIIX', 'IIIIIXIIIII
X', 'IIIIIIXIIIIX', 'IIIIIIIIIIXX'],
              coeffs=[1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
1.+0.j, 1.+0.j,
 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j,
 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j]), coe
ff=1)
```

In [51]:
```
"""
Calculating exact eigenvalue
"""
from qiskit.algorithms import NumPyEigensolver

# find the first three (k=3) eigenvalues
exact_solver = NumPyEigensolver(k=3)
exact_result = exact_solver.compute_eigenvalues(ham)
print(exact_result.eigenvalues)

# Save ground state energy for later
gs_energy = np.round(exact_result.eigenvalues[0], 4)
```

```
[-18.         -18.         -16.96132284]
```

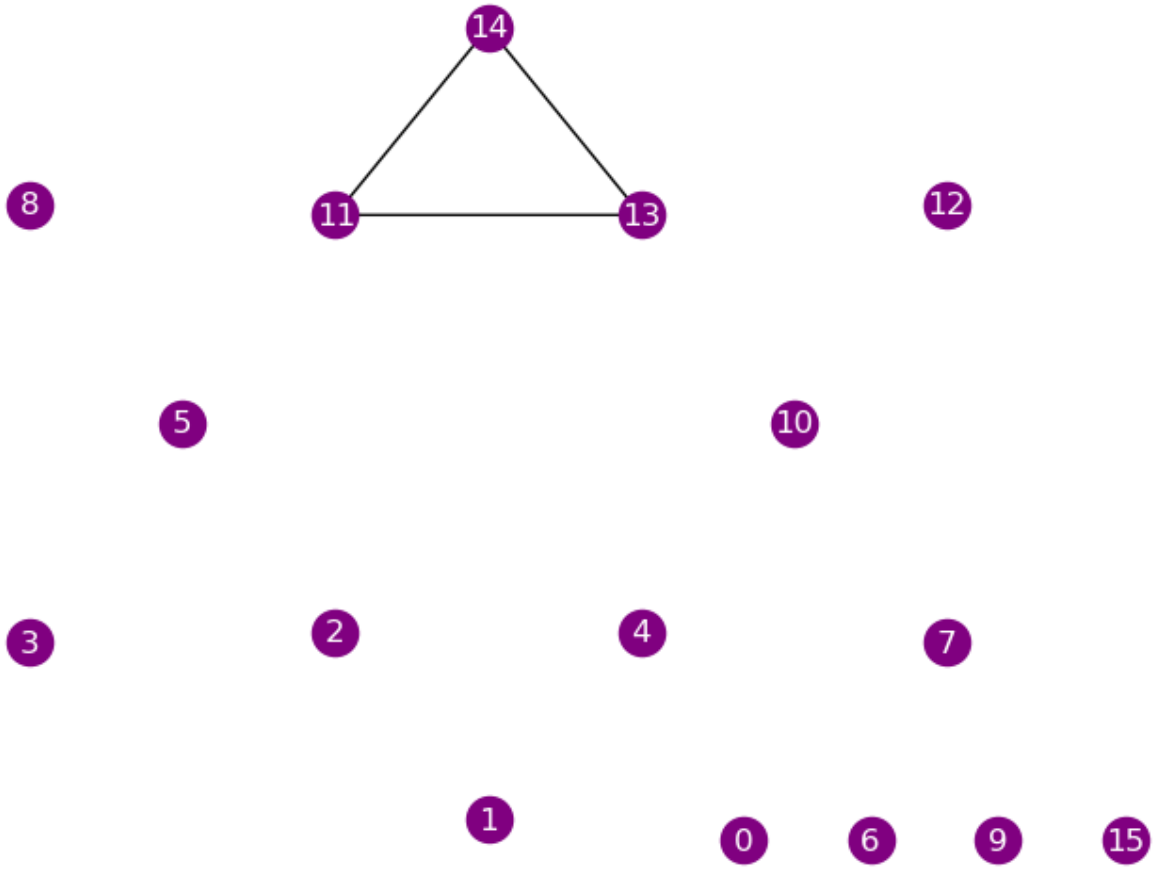
In [52]:
```
"""
Importing packages, IBMQ and the transpiler
"""

from qiskit import QuantumCircuit, transpile
from qiskit.circuit import Parameter

from qiskit import IBMQ
```

In [53]:
```
"""
Enter account info
"""

# Doc for loading IBMQ account https://quantum-computing.ibm.com/lab/docs/iql/m
#TOKEN='Please enter your own token here'
#IBMQ.save_account(TOKEN)
IBMQ.load_account() # Load account from disk
provider = IBMQ.get_provider(hub='ibm-q-community', group='ibmquantumawards', p

# Real backend; needed for transpilation later on
guadalupe = provider.get_backend("ibmq_guadalupe")
```

```
ibmqfactory.load_account:WARNING:2023-04-14 14:59:54,080: Credentials are alre
ady in use. The existing account in the session will be replaced.
```

In [54]:

```
"""
Specify connections between nodes. A key part of our strategy is to use a subse
and then to use the hexagonal symmetry to estimate the ground state.
"""

# Kagome unit cell
num_qubits = 16
# Edge weight
t = 1.0

# Generate graph of kagome unit cell
# Start by defining all the edges
graph_16 = rx.PyGraph(multigraph=False)
graph_16.add_nodes_from(range(num_qubits))


#Specify edge list
edge_list = [
    (11, 14, t),
    (14, 13, t),
    (11, 13, t)
]

# Generate graph from the list of edges
graph_16.add_edges_from(edge_list)

# Make a Lattice from graph
kagome_unit_cell_16 = Lattice(graph_16)

# Draw Lattice and include labels to check we exclude the right spins
# Specify node locations for better visualizations
plt.figure()
kagome_pos = {0:[1,-1], 6:[1.5,-1], 9:[2,-1], 15:[2.5,-1],
              1:[0,-0.8], 2:[-0.6,1], 4:[0.6,1], 10:[1.2,3],
              13:[0.6,5], 11:[-0.6,5], 5:[-1.2,3], 3:[-1.8,0.9],
              8:[-1.8,5.1], 14:[0,6.8], 7:[1.8,0.9], 12:[1.8,5.1]}
kagome_unit_cell_16.draw(style={'with_labels':True, 'font_color':'white', 'node
plt.savefig('kagome-unit-cell-numbers.png',dpi=600)
```



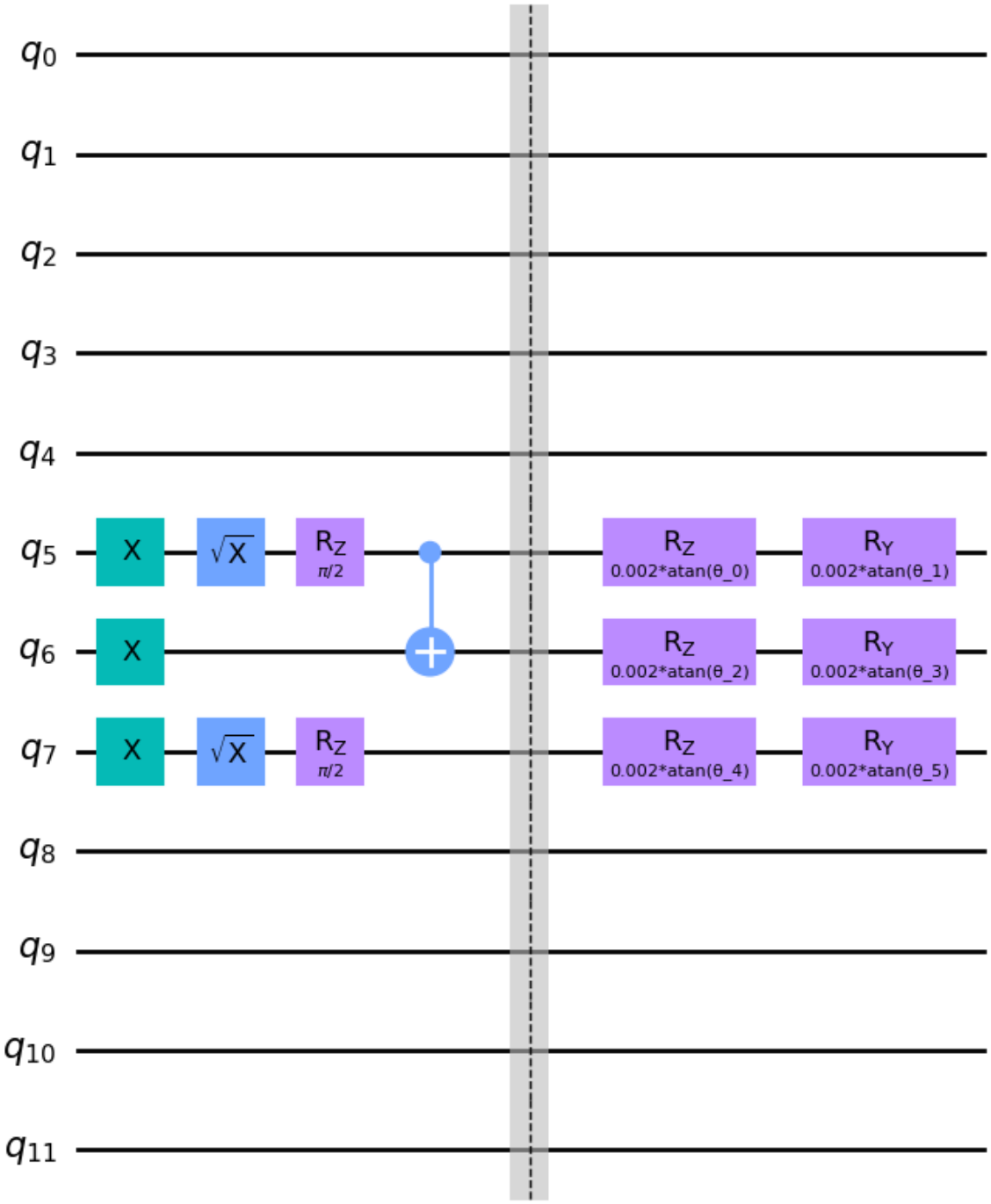
In [55]:
```
"""
Building Hamiltonian from graph edges. A factor of six has been included in ham
exploit
"""
# Build Hamiltonian from graph edges
heis_16 = HeisenbergModel.uniform_parameters(
    lattice=kagome_unit_cell_16,
    uniform_interaction=t,
    uniform_onsite_potential=0.0,  # No singe site external field
)

# Map from SpinOp to qubits just as before.
log_mapper = LogarithmicMapper()
ham_16 = 6*4 * log_mapper.map(heis_16.second_q_ops().simplify())
# Print Hamiltonian to check it's what we expect:
# 18 ZZ, 18 YY, and 18 XX terms over 16 qubits instead of over 12 qubits
ham_16
```

Out[55]:
```
PauliSumOp(SparsePauliOp(['IZZIIIIIIIIIIIII', 'IZIIZIIIIIIIIIII', 'IIZIZIIIIII
IIIII', 'IYYIIIIIIIIIIIII', 'IYIIYIIIIIIIIIII', 'IIYIYIIIIIIIIIII', 'IXXIIIIII
IIIIIII', 'IXIIXIIIIIIIIIII', 'IIXIXIIIIIIIIIII'],
              coeffs=[6.+0.j, 6.+0.j, 6.+0.j, 6.+0.j, 6.+0.j, 6.+0.j, 6.+0.j,
6.+0.j, 6.+0.j]), coeff=1)
```

In [56]:
```
from qiskit.circuit.library import EfficientSU2
from qiskit.tools.visualization import circuit_drawer

"""
Construct the ansatz from scratch.
```

```
Our ansatz consists of two sections demarcated by a barrier. There is an initia
prepared. This part of the circuit is motivated by the analytical solution of a
state is a bell state.

The second section consists of single qubit rotations which can correct for sys

In our tunable phase gates, we use the arcan of the parameters instead of the p
the range of the parameters and can improve convergence.

"""

# Build a custom ansatz from scratch
import random
from qiskit.tools.visualization import circuit_drawer
ansatz_custom = QuantumCircuit(12)

#Use the parity operator
ansatz_custom.x(5)
ansatz_custom.x(6)
ansatz_custom.x(7)

ansatz_custom.sx(5)
ansatz_custom.rz(np.pi/2,5)

ansatz_custom.cx(5,6)

ansatz_custom.sx(7)
ansatz_custom.rz(np.pi/2,7)

ansatz_custom.barrier()


#apply single qubit phase gates to the qubits of the triangle.
r=0.001
for k in range(5,8):
    param_z=Parameter('θ_' + str(2*(k-5)))
    param_y=Parameter('θ_' + str(2*(k-5)+1))
    ansatz_custom.rz(2*r*np.arctan(param_z),k)
    ansatz_custom.ry(2*r*np.arctan(param_y),k)


ansatz_custom.draw(fold=350)

circuit_drawer(ansatz_custom, output='mpl', filename='custom_ansatz.png')
```

Out[56]:



In [57]:
```
print(['the depth is',str(ansatz_custom.depth())])
```

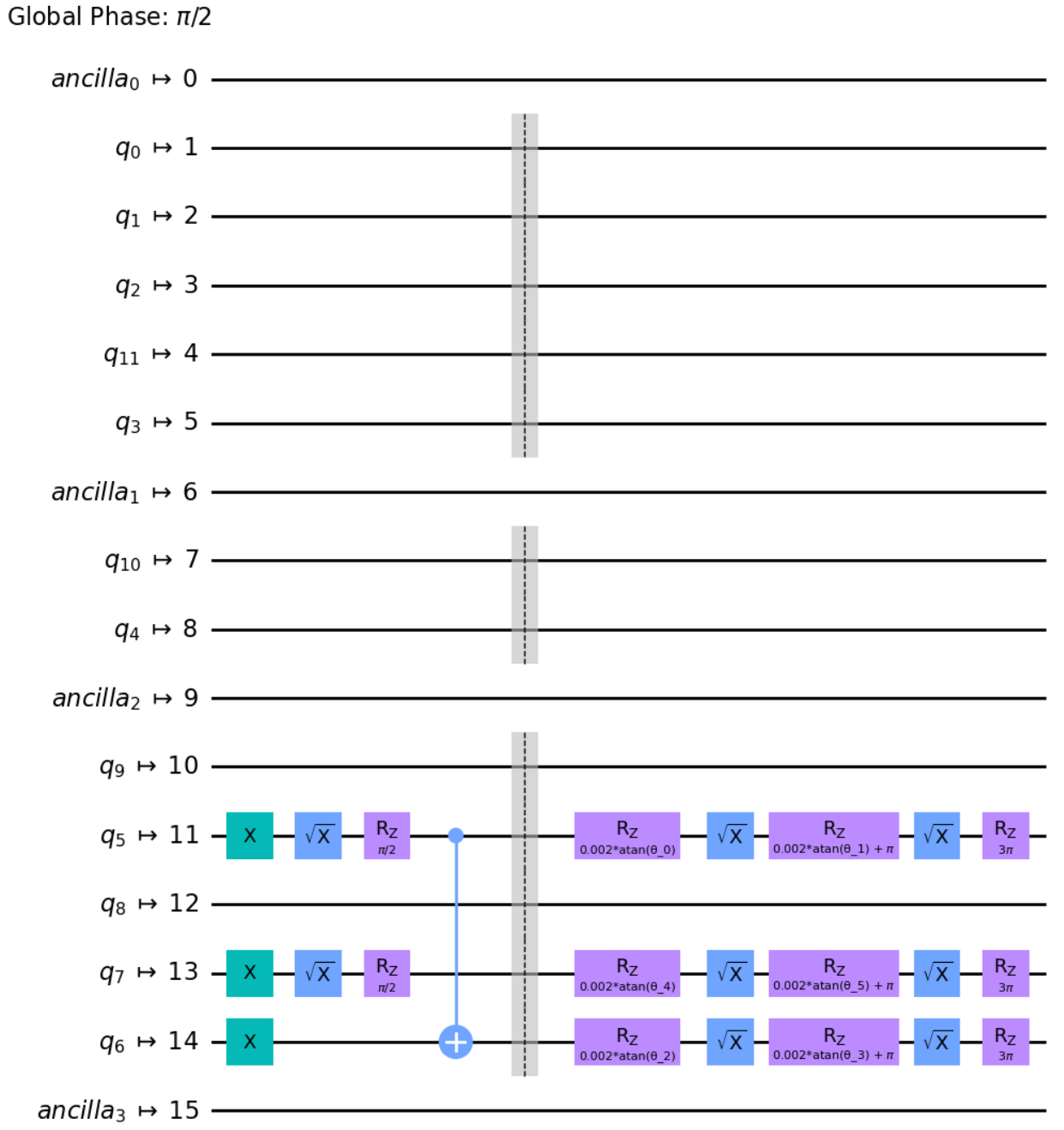
```
['the depth is', '6']
```

In [58]:
```
"""
Transpile the circuit
"""

# Force anstaz to be applied to qubits in the heavy hex.
# Avoid the outer qubits 0, 6, 9, and 15 which we accounted for in the lattice
q_layout = [1, 2, 3, 5, 8, 11, 14, 13, 12, 10, 7, 4]
ansatz_opt = transpile(ansatz_custom, backend=guadalupe, initial_layout=q_layou


print('number and type of gates in the cirucit:', ansatz_opt.count_ops())
```

```
print('number of parameters in the circuit:', ansatz_opt.num_parameters)
ansatz_opt.draw(fold=300)

circuit_drawer(ansatz_opt, output='mpl', filename='custom_ansatz_opt.png')
```

```
number and type of gates in the cirucit: OrderedDict([('rz', 11), ('sx', 8),
('x', 3), ('cx', 1), ('barrier', 1)])
number of parameters in the circuit: 6
```

Out[58]:



In [59]:

```
print(['the optimized depth is',str(ansatz_opt.depth())])
```

```
['the optimized depth is', '9']
```

In [60]:

```
"""
Specify the optimizer we want to use.

We chose the perturbation to minimize the error associated with the estimation
For more information about how this parameter was chosen please see the documer
```

```
The learning rate was selected to be very small because we are already close to
variational parameters are set to zero.
"""

from qiskit.algorithms.optimizers import SPSA

optimizer = SPSA(maxiter=200,perturbation=0.05,learning_rate=0.00002)
```

In [61]:
```
"""
Specify the VQE Program.

We choose our initial ansatz parameters to be close to zero because that corres
"""

from qiskit.algorithms import MinimumEigensolver, VQEResult

# Define a custome VQE class to orchestra the ansatz, classical optimizers,
# initial point, callback, and final result
class CustomVQE(MinimumEigensolver):

    def __init__(self, estimator, circuit, optimizer, callback=None):
        self._estimator = estimator
        self._circuit = circuit
        self._optimizer = optimizer
        self._callback = callback

    def compute_minimum_eigenvalue(self, operators, aux_operators=None):

        # Define objective function to classically minimize over
        def objective(x):
            # Execute job with estimator primitive
            job = self._estimator.run([self._circuit], [operators], [x])
            # Get results from jobs
            est_result = job.result()
            # Get the measured energy value
            value = est_result.values[0]
            # Save result information using callback function
            if self._callback is not None:
                self._callback(value)
            return value

        # Select an initial point for the ansatzs' parameters
        np.random.seed(seed=1)
        x0 =0.0*np.random.randn(self._circuit.num_parameters)

        # Run optimization
        res = self._optimizer.minimize(objective, x0=x0)

        # Populate VQE result
        result = VQEResult()
        result.cost_function_evals = res.nfev
        result.eigenvalue = res.fun
        result.optimal_parameters = res.x
        return result
```

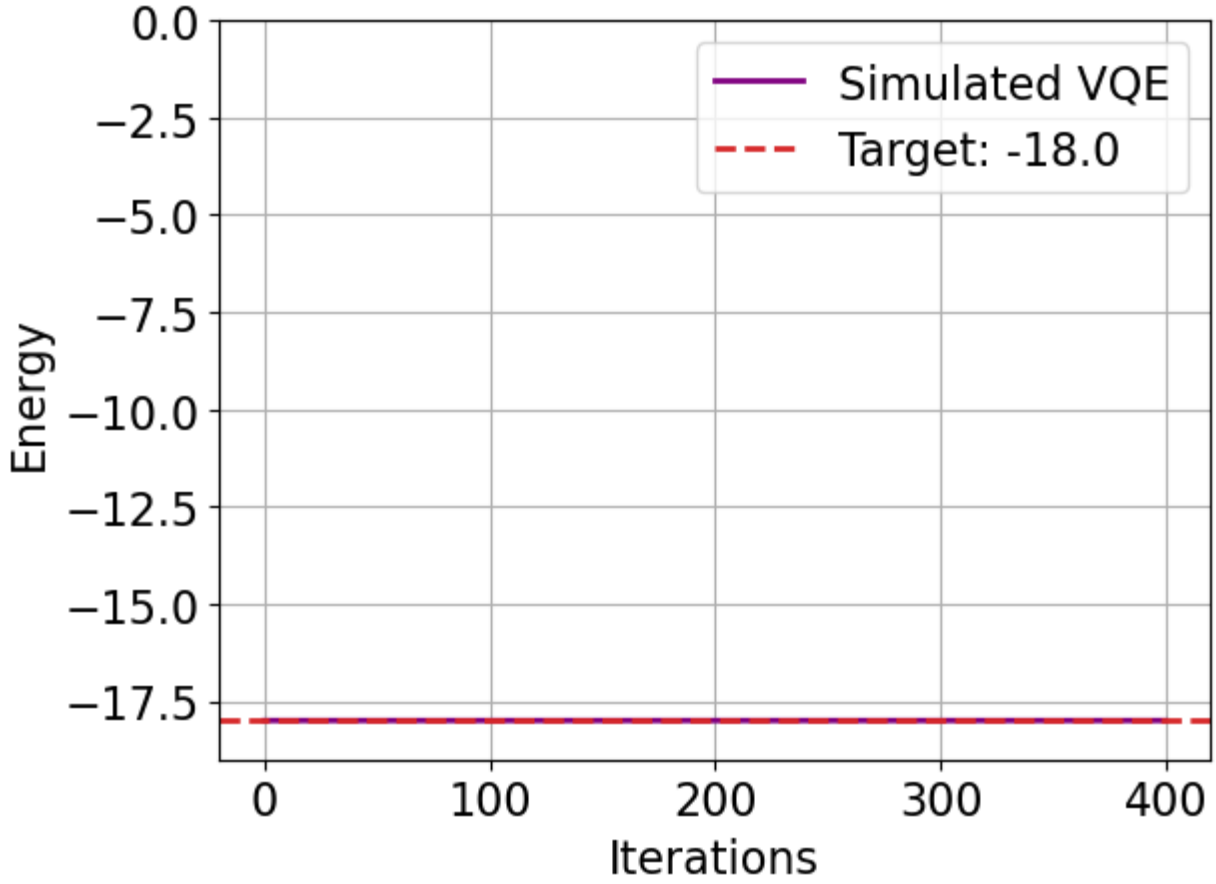
In [62]:
```
# Define a simple callback function
intermediate_info = []
```

```
def callback(value):
        intermediate_info.append(value)
```

In [63]:
```
from qiskit.primitives import Estimator
import time

"""
Execute simulated VQE
"""

# Define instance of qiskit-terra's Estimator primitive
num_shots=99999
options = {'shots': num_shots}
estimator = Estimator([ansatz_opt], [ham_16],options=options)


# Setup VQE algorithm
custom_vqe = CustomVQE(estimator, ansatz_opt, optimizer, callback=callback)

# Run the custom VQE function and monitor execution time
start = time.time()
result = custom_vqe.compute_minimum_eigenvalue(ham_16)
end = time.time()

print(result)
print(f'execution time (s): {end - start:.2f}')
```

```
{   'aux_operator_eigenvalues': None,
    'cost_function_evals': 400,
    'eigenstate': None,
    'eigenvalue': -17.99999999917281,
    'optimal_circuit': None,
    'optimal_parameters': array([-3.96467245e-08, -1.84657586e-08,  6.44449098
e-08,  5.28876451e-09,
        7.18135449e-09, -8.96707486e-09]),
    'optimal_point': None,
    'optimal_value': None,
    'optimizer_evals': None,
    'optimizer_result': None,
    'optimizer_time': None}
execution time (s): 3.67
```

In [64]:
```
"""
Plot and save results
"""
#Save info
pickle.dump(intermediate_info, open('intermediate_info.pkl', 'wb'))

plt.figure()
plt.plot(intermediate_info, color='purple', lw=2, label='Simulated VQE')
plt.ylabel('Energy')
plt.ylim([-19,0])
plt.xlabel('Iterations')
# Exact ground state energy value
plt.axhline(y=gs_energy, color="tab:red", ls="--", lw=2, label="Target: " + str
plt.legend()
plt.grid()
plt.savefig('VQE_SIM.png',dpi=600)
```



In [65]:
```
"""
Compute the relative error.
"""

def rel_err(target, measured):
    return abs((target - measured) / target)

# Compute the relative error between the expected ground state energy and the V
rel_error = rel_err(gs_energy, result.eigenvalue)



print(f'Expected ground state energy: {gs_energy:.10f}')
print(f'Computed ground state energy: {result.eigenvalue:.10f}')
print(f'Relative error: {rel_error:.8f}')
```

```
Expected ground state energy: -18.0000000000
Computed ground state energy: -17.9999999992
Relative error: 0.00000000
```

In [ ]:
```
"""
Moving onto execution on Guadalupe
"""
```

In [ ]:
```
"""
Specify parameters for execution on Guadalupe.

We use the max number of shots, resilience level one and optimization level one
"""
```

```
from qiskit_ibm_runtime import (QiskitRuntimeService, Session,Options,
                                Estimator as RuntimeEstimator)
from qiskit import Aer

service = QiskitRuntimeService(channel='ibm_quantum',
    instance='ibm-q-community/ibmquantumawards/open-science-22')
options = Options()
options.execution.shots=99999
options.resilience_level = 1
options.optimization_level = 1
backend = 'ibmq_guadalupe'
```

In [ ]:
```
"""
Call back function for the real run
"""
# Define a simple callback function
intermediate_info_real_backend = []
def callback_real(value):
        intermediate_info_real_backend.append(value)
```

In [ ]:
```
"""
Specify time out handler
"""

import signal, time

from qiskit_ibm_runtime import Estimator, Session
from qiskit.providers import JobStatus

def timeout_handler(signum, frame):
    raise Exception('Iteration timed out')

class RetryEstimator(Estimator):
    """RuntimeRetryEstimator class.

    This class inherits from Qiskit IBM Runtime's Estimator and overwrites its
    a maximum of 'max_retries' consecutive times, if it encounters one of the f

    * An Estimator error (in this case "Job.ERROR" is printed, and the job is c
    * A timeout error where the job either remains running or completes but doe
      than 'timeout' (in this case the job is cancelled by the patch and "Job.C
    * A creation error, where the job fails to be created because connection is
      quantum computer (in this case "Failed to create job." is printed). If th
      to a new Session (to be handled with care! also, this will unfortunately
    """

    def __init__(self, *args, max_retries: int = 5, timeout: int = 3600, **kwar
        super().__init__(*args, **kwargs)
        self.max_retries = max_retries
        self.timeout = timeout
        self.backend = super().session._backend
        signal.signal(signal.SIGALRM, timeout_handler)

    def run(self, circuits, observables, parameter_values, **kwargs):
        result = None
        for i in range(self.max_retries):
            try:
                job = super().run(circuits, observables, parameter_values, **kw
                while job.status() in [JobStatus.INITIALIZING, JobStatus.QUEUEI
```

```
                    time.sleep(5) # Check every 5 seconds whether job status ha
                signal.alarm(self.timeout) # Once job starts running, set timeo
                result = job.result()
                if result is not None:
                    signal.alarm(0) # reset timer
                    return job
            except Exception as e:
                print("\nSomething went wrong...")
                print(f"\n\nERROR MESSAGE:\n{e}\n\n")
                if 'job' in locals(): # Sometimes job fails to create
                    print(f"Job ID: {job.job_id}. Job status: {job.status()}.")
                    if job.status() not in [JobStatus.DONE, JobStatus.ERROR, Jo
                        job.cancel()
                else:
                    print("Failed to create job.")
                print(f"Starting trial number {i+2}...\n")
                print(f"Creating new session...\n")
                signal.alarm(0) # reset timer
                super().session.close()
                self._session = Session(backend=self.backend)
        if result is None:
            raise RuntimeError(f"Program failed! Maximum number of retries ({se
```

In [ ]:
```
"""
Execute on Guadalupe
"""

start = time.time()
with Session(service=service, backend=backend) as session:
    # Prepare extended primitive
    rt_estimator = RetryEstimator(session=session,options=options)
    # set up algorithm
    custom_vqe = CustomVQE(rt_estimator, ansatz_opt, optimizer, callback=callba
    # run algorithm
    result = custom_vqe.compute_minimum_eigenvalue(ham_16)
end = time.time()
print(f'execution time (s): {end - start:.2f}')
```

In [ ]:
```
"""
Plot the energies
"""

#calculate the cumulative mean
means = np.cumsum(intermediate_info_real_backend) / np.arange(1, len(intermedia

plt.figure()
plt.plot(means, color='purple', lw=2, label='VQE')
plt.ylabel(r'$\langle Energy \rangle$')
plt.xlabel('Iterations')
plt.axhline(y=gs_energy, color="tab:red", ls="--", lw=2, label="Target: " + str
plt.legend()
plt.grid()
plt.savefig('vqe_real.png',dpi=600)
plt.show()
```

In [ ]:
```
"""
Show final relative error.
"""
```

```
# Compute the relative error between the expected ground state energy and the m
computed_gse = np.mean(intermediate_info_real_backend)#Take mean for best perfo

print(f'Expected ground state energy: {gs_energy:.8f}')
print(f'Computed ground state energy: {computed_gse:.8f}')
print(f'Relative error: {100 * rel_err(gs_energy, computed_gse):.8f} %')
```

This page is intentionally blank.

INITIAL DISTRIBUTION

84310	Technical Library/Archives	(1)
71730	S. Crowe	(1)
55240	R. Rodriguez	(1)
NRL	J. Stenger	(1)
SWC	J. Adajar	(1)
SWC	J. Afalla	(1)
72110	D. Rees	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

This page is intentionally blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
January 2024	Final	

4. TITLE AND SUBTITLE

Analysis of Heisenberg Models Using Variational Quantum Eigensolvers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHORS

Sean T. Crowe
Ramiro Rodriguez
NIWC Pacific

John Stenger
Naval Research Laboratory

Jaecob Adajar
Jarrod Afalla
Southwestern College

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NIWC Pacific
53560 Hull Street
San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-3330

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Innovative Science and Engineering
NIWC PACIFIC
53560 HULL ST, SAN DIEGO CA 92152-5001

10. SPONSOR/MONITOR'S ACRONYM(S)

NISE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this report, we analyze several Heisenberg models using the IBM Quantum computer Guadalupe. Variational eigensolving techniques are used, and their respective ingredients are discussed and motivated.

15. SUBJECT TERMS

Quantum Computer; Variational Eigensolving; Heisenberg models

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Sean T. Crowe
U	U	U	SAR	46	**19b. TELEPHONE NUMBER** *(Include area code)* 631-317-5016

This page is intentionally blank.

This page is intentionally blank.

Approved for public release. Distribution is unlimited.

Naval Information Warfare Center (NIWC) Pacific
San Diego, CA 92152-5001