Variational Quantum Eigensolver (VQE) in Qiskit

Riley Peterlinz

January 20th, 2022

Goal and Structure

Goal: To show how to implement and use VQE in Qiskit.

Structure:

- Part 1: Brief Review of VQE
- Part 2: Implementing VQE in Qiskit
- Part 3: Some Experimental Results

Part 1: Brief Review of VQE

Goal: We want to find the ground state energy of some Hamiltonian.

Inputs: Hamiltonian H and Guessed Wavefunction (Ansatz) with Numerical Parameters

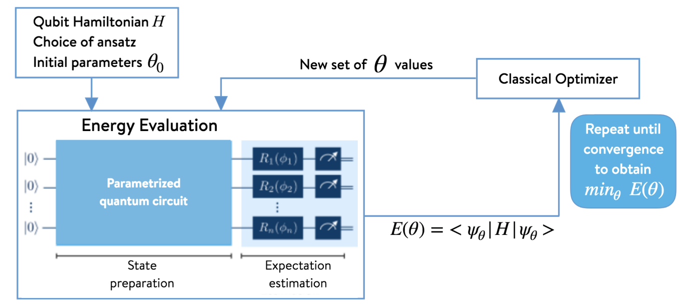


Image source: OpenQEMIST

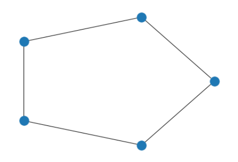
Part 2: Implementing VQE

Implementing VQE: Hamiltonian

Used an Periodic Ising Hamiltonian on a Lattice

$$H = \sum_{i,j}^{n} J_{ij} Z_i Z_j + \sum_{i}^{n} C_i X_i \quad J_{ij}, C_i \in \mathbb{R}$$

- $n = 5$



$$-ZZIII - IZZII - IIZZI - IIIZZ - ZIIIZ + XIIII + IXIII + IIXII + IIIXI + IIIIX$$

Implementing VQE: VQE Package

```
from qiskit.algorithms import VQE
```

VQE is a function that must be initiated with two **Parameters**:

- Simulator to run the Algorithm
- Ansatz

Once initiated, the function takes Hamiltonian as its input and returns an approximate Groundstate Energy amongst other data

Building a VQE: Ansatz

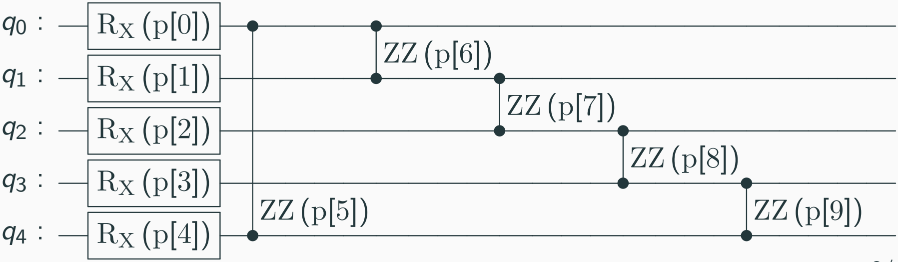
The Ansatz is a **Circuit**

```
from qiskit.circuit.library import TwoLocal
ansatz = TwoLocal(
        rotation_blocks=["rx"],
        entanglement_blocks="rzz",
        entanglement="circular",
        skip_final_rotation_layer = skip,
        reps = 1,
        parameter_prefix="p",)
```

$$-ZZIII - IZZII - IIZZI - IIIZZ - ZIIIZ + XIIII + IXIII + IIXII + IIIXI + IIIIX$$

Building a VQE: Ansatz

```
            entanglement="circular",
            skip_final_rotation_layer = skip,
            reps = 1,
ansatz.num_qubits = 5
ansatz.decompose().draw("latex")
```



Implementing VQE: Using the VQE

Now, we can input some Hamiltonian represented as a **PauliSumOp** into our VQE!

```
vqe_solver = VQE(
    ansatz = ansatz,
    quantum_instance = QuantumInstance(
        Aer.get_backend("aer_simulator_statevector")))

ham = PauliSumOp(...)
result = vqe_solver.compute_minimum_eigenvalue(ham)
```

Implementing VQE: Results

```
print(result)
{   'aux_operator_eigenvalues': None,
    'cost_function_evals': 6,
    'eigenstate': array([...]),
    'eigenvalue': (9.7e-17+0j),
    'optimal_parameters': { ... },
    'optimal_point': array([...]),
    'optimal_value': 9.7e-17,
    'optimizer_evals': None,
    'optimizer_time': 0.351621150970459}
```

Part 3: Experimental Results

Experimental Results: Method and Data

The Goal of the Experiments was to see how accurate the VQE's Ground State compares to the **True Ground State**.

Each Plot is the **difference** between the VQE and True Ground State (smaller is better)

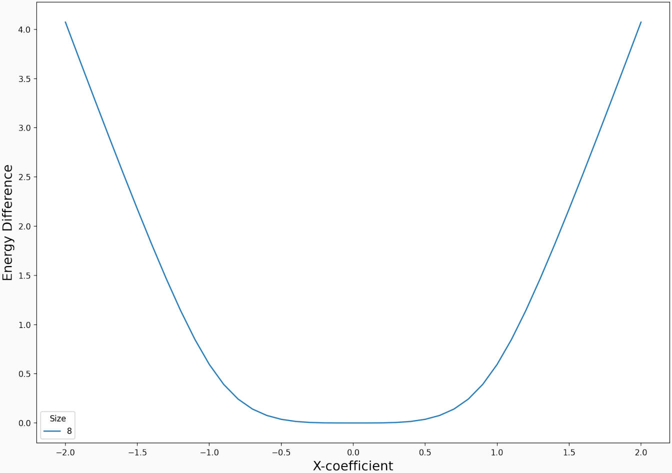
$$H = \sum_{i,j}^{n} J_{ij} Z_i Z_j + \sum_{i}^{n} C_i X_i \quad J_{ij}, C_i \in \mathbb{R}$$

Experiments

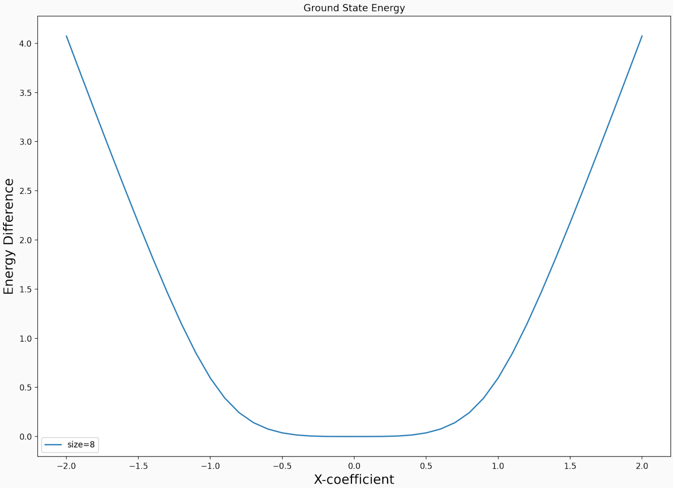
- Varying C_i from $[-2.0, 2.0]$ in increments of 0.1
- Number of Rounds in the Ansatz from $1 - 5$

Ferromagnetic ($J_{ij} = -1$) and Anti-Ferromagnetic ($J_{ij} = 1$)

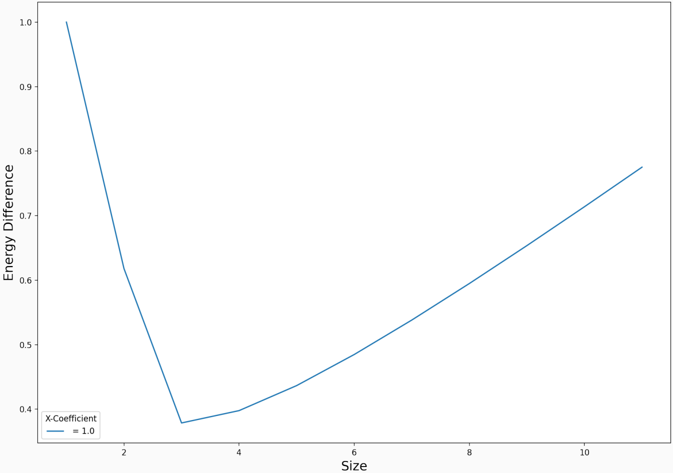
Varied X-Coefficient: Ferromagnetic Ground State Energy Difference



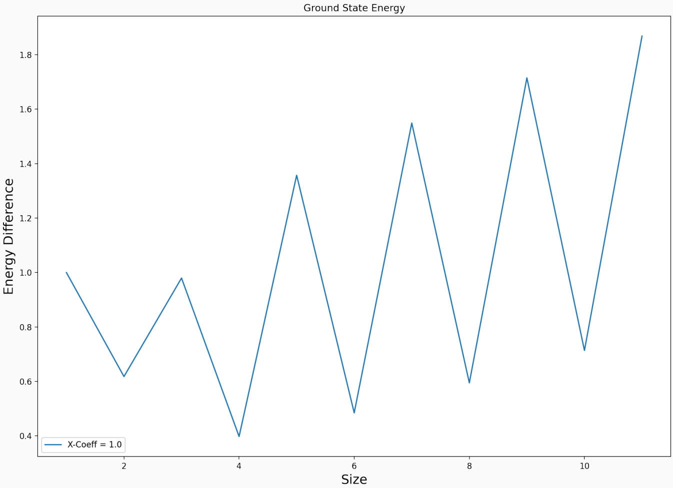
Varied X-Coefficient: Anti-Ferromagnetic Ground State Energy Difference



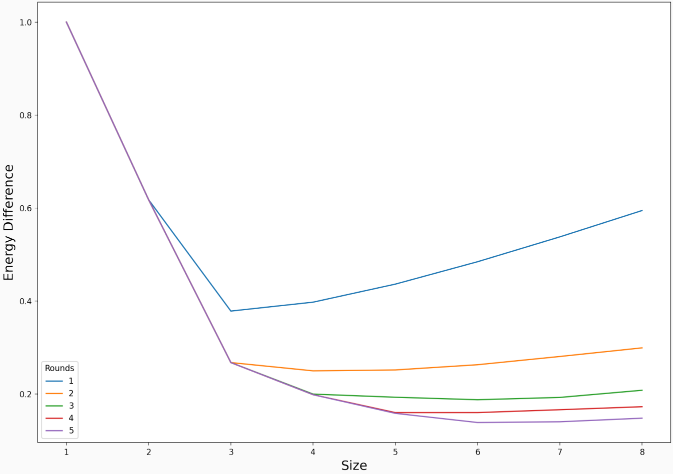
Increasing Size: Ferromagnetic Ground State Energy Difference



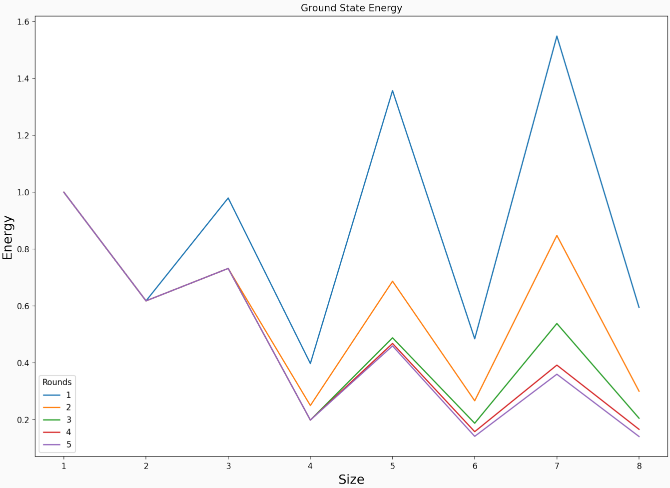
Increasing Size: Anti-Ferromagnetic Ground State Energy Difference



Number of Rounds: Ferromagnetic Ground State Energy Difference



Number of Rounds: Anti-Ferromagnetic Ground State Energy Difference



Gratitude

Thank you for listening!

Special thanks to Ani!