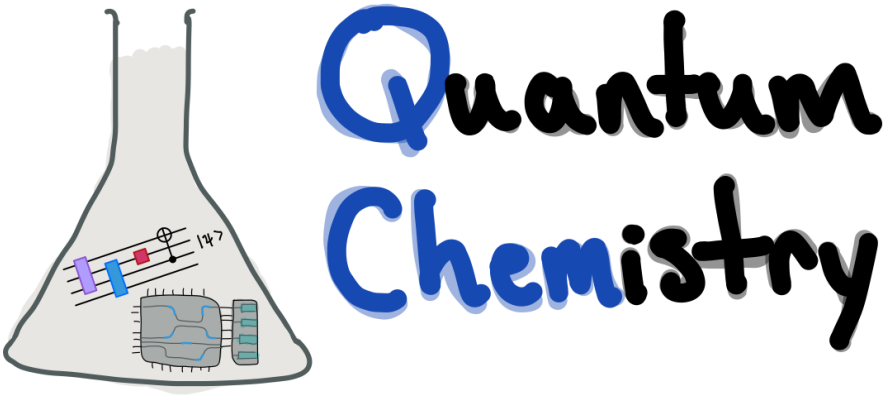

Quantum Chemistry: Optimizing Measurements [200 points]

Version: 1

Quantum Chemistry

Numerical techniques for determining the structure and chemical properties of molecules are a juggernaut area of research in the physical sciences. *Ab initio* methods like density functional theory have been a staple method in this field for decades, but have been limited in scalability and accuracy. As such, chemistry applications and problems are desirable candidates for demonstrating a quantum advantage.

It is therefore no surprise that quantum chemistry is one of the leading application areas of quantum computers. In the **Quantum Chemistry** category, you will be using PennyLane's core quantum chemistry functionalities to become familiarized with concepts and tools developed in this sub-field of quantum computing, like mapping molecular Hamiltonians to qubit Hamiltonians and quantum gates that preserve the electron number. Beyond these five questions in this category, there is a plethora of informative tutorials on the PennyLane website that will boost your understanding of topics that we will cover in this category. Let's get started!

Problem statement [200 points]

One of the fundamental algorithms to determine the electronic structure of molecules is the Variational Quantum Eigensolver (VQE). To carry out this algorithm, we must know the energy of the molecule given an electronic configuration or structure. This structure is encoded in a quantum circuit with a set

of initial gates and, to calculate the energy of that configuration, we need the expectation value of the Hamiltonian defined by the molecule.

However, given an arbitrary Hamiltonian, we cannot calculate its expectation value directly: we usually decompose the Hamiltonian into a linear combination of *Pauli words* (e.g., `qml.PauliZ(0) @ qml.PauliX(1) @ qml.PauliY(2)`). In this way, we can calculate the expected value of each of these Pauli words, add them up, and obtain the expectation value of the Hamiltonian.

A Pauli word is defined by a tensor product of Pauli operators I, X, Y and Z applied on different qubits. The identity operator I is used in positions where no measurement needs to be made. Suppose, for example, that the electronic configuration is defined by a circuit with 4 qubits and we are interested in calculating the expectation value of the operator $Z(0) \otimes Y(1) \otimes I(2) \otimes I(3)$ (the argument in parentheses represents the qubit where we want to apply this operator). Then what we must do is perform a series of executions of the circuit, measuring in the Z axis on qubit 0 and on the Y axis on qubit 1. A diagrammatic example of this can be seen in Figure 1.

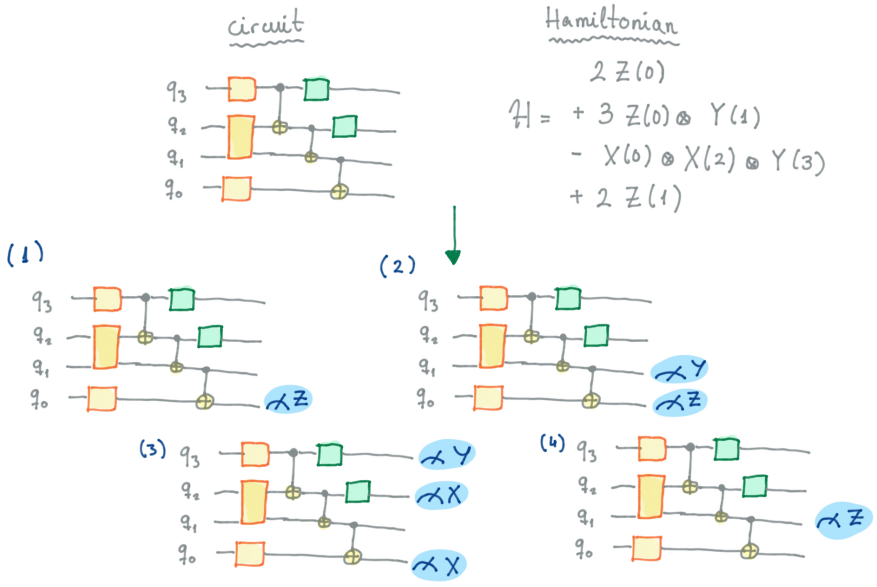

Figure 1: Decomposition

In this case, we have a Hamiltonian that we have divided into 4 Pauli words. So, given an electronic configuration defined by the initial state, we will have to run four different circuits: one for each operator in the Pauli word. The number of circuits required in a general Hamiltonian grows like $O(N^4)$, where N is the number of qubits. Therefore, it is desirable to look for techniques that reduce

this scaling. In this challenge we will work with a very simple technique. There are two rules we can play with:

1. If two circuits do not intersect in any of their measurement qubits, they could be executed at the same time (see Figure 2).

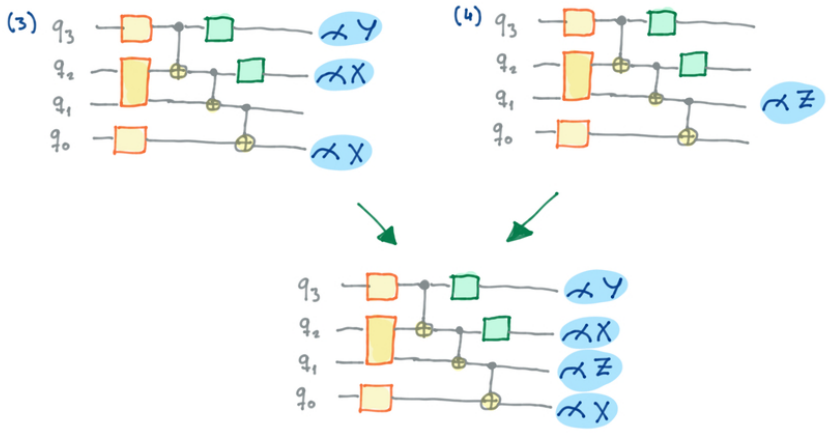

Figure 2: Rule 1

With this rule, by processing the output of the final circuit we can obtain the equivalent results to the two initial circuits, thus reducing the number of executions required.

2. If two circuits intersect in some of their measurement qubits but the measurement operators also coincide in all of them, they can be joined (see Figure 3).

You are challenged with compressing the number of measurements needed for some given Hamiltonians using these two rules. In the provided template called `optimizing_measurements_template.py`, there are a few functions that you need to complete:

- `check_simplification`: Given two Pauli words, you might be able to obtain the expected value of each of them by running a single circuit. This function checks whether or not this is possible.
- `join_operators`: Given two Pauli words whose expected values can be calculated by running a single circuit, this function returns the operator corresponding to the union of both of them.
- `compression_ratio`: This function calculates the compression ratio of this procedure. I.e., the reduction factor for how many circuit evaluations need to be done using the previously mentioned rules versus a naive

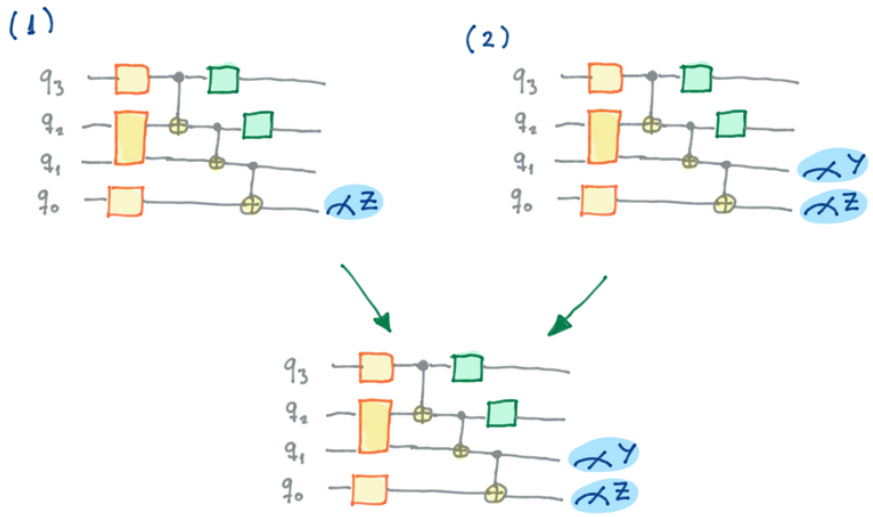

Figure 3: Rule 2

implementation with no reductions,

$$r = 1 - \frac{\text{nº final observables}}{\text{nº initial observables}}.$$

Input

- `list`: A list containing the number of wires required to define a quantum circuit and Pauli operators. The Pauli operators are grouped to create Pauli words. These Pauli words are then used to define the Hamiltonian of interest. For example:

```
[
    ["Z", "I", "I", "I"],
    ["Z", "Y", "I", "I"],
    ["X", "I", "X", "Y"],
    ["I", "Z", "I", "I"]
]
```

Output

- `float`: The compression ratio.

Acceptance Criteria

In order for your submission to be judged as "correct":

- The outputs generated by your solution when run with a given `.in` file must match those in the corresponding `.ans` file to within the `0.0001` tolerance specified below. To clarify, your answer must satisfy

$$\text{tolerance} \geq \left| \frac{\text{your solution} - \text{correct answer}}{\text{correct answer}} \right|.$$

- Your solution must take no longer than the `60s` specified below to produce its outputs.

WARNING: Don't modify the code outside of the `# QHACK #` markers in the template file, as this code is needed to test your solution. Do not add any print statements to your solution, as this will cause your submission to fail.

Specs
Tolerance: **0.0001**
Time limit: **60 s**

Version History

Version 1: Initial document.