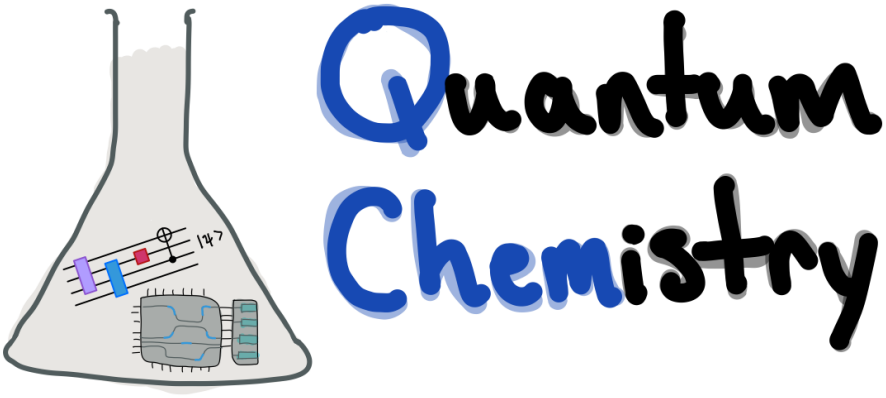

Quantum Chemistry: Particle Conservation [100 points]

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 [100 points]

In quantum chemistry, we represent the states of electrons in an atom by labelling the occupation levels. This notation is known as the Jordan-Wigner representation. We write 1 when an energy level is occupied and 0 when it is not. Consider the state

$|1001\rangle$.

The first level is represented by the first two states in the tensor product (the purple states) since two electrons can be in one energy level. The second energy level corresponds to the second two states (the green states), and so on.

When we perform operations on atoms, we expect the number of electrons to remain constant. If we use the Jordan-Wigner representation, only certain unitaries are allowed: those that preserve the *Hamming weight* of the state — the number of 1s in the tensor product. Your task is to determine whether a given circuit preserves the Hamming weight of any arbitrary input state.

To do this, we evaluate a given circuit for every element of the computational basis. We can then calculate the Hamming weight of the output to determine if the Hamming weight is preserved for each element of the computational basis.

The provided template `particle_conservation_template.py` contains a few functions that you need to complete:

- `binary_list`: Converts a number m to its binary representation. For example, if we work with four qubits, $m = 7$ would give $[0, 1, 1, 1]$.
- `basis_states`: Creates all elements of the computational basis for n qubits (2^n elements). The result of this function lets us set the initial state of a quantum circuit to be any of the computational basis states.
- `is_particle_preserving`: This is where you will determine if a given circuit conserves the particle number. You will loop over all states from `basis_states`, calculate the resulting state via `circuit(state)` (this function is provided to you), and compare the Hamming weights to see if the number of particles is conserved. Output `True` (`False`) if the circuit is (is not) particle-conserving.

Input

- `list`: A list containing PennyLane operations and arguments required to define those operations for use in a quantum circuit.

Output

- `bool`: `True` if the circuit preserves the number of particles, `False` if it doesn't.

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.

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

You can test your solution by passing the `#.in` input data to your program as stdin and comparing the output to the corresponding `#.ans` file:

```
python3 {name_of_file}.py < 1.in
```

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
Time limit: **60 s**

Version History

Version 1: Initial document.