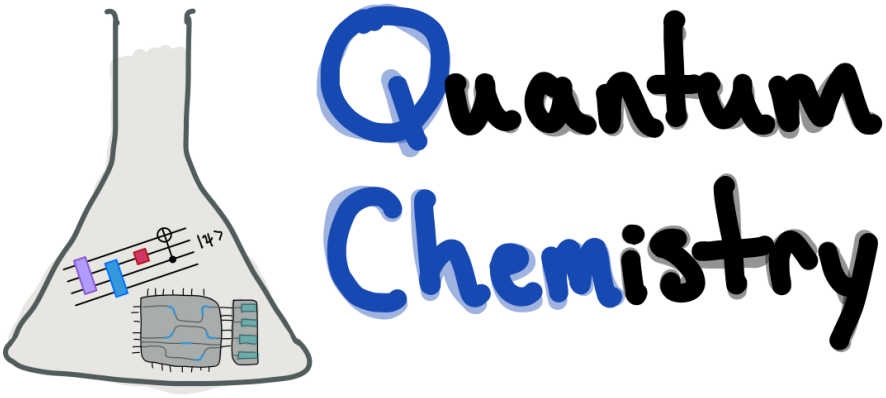

Quantum Chemistry: Triple Givens [400 points]

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 [400 points]

You will once again be dealing with Givens rotations in this problem. Briefly, Givens rotations are unitary operators that are especially friendly for quantum chemistry applications since they conserve the Hamming weight (or particle number). In a quantum chemistry context, they map N-electron states to other N-electron states (they neither create nor annihilate electrons). There is a succinct PennyLane demo on Givens rotations here. Go check it out!

In this challenge, you will be implementing a triple-excitation Givens rotation operator. The method of implementation we recommend that you follow will be to create, by hand, the matrix representation of a triple-excitation Givens rotation. Then, with this matrix representation in hand, mould it into a PennyLane unitary with `qml.QubitUnitary`. Just like the other already-implemented Givens rotations in PennyLane (i.e. `qml.SingleExcitation` and `qml.DoubleExcitation`), the triple-excitation Givens rotation takes one angle as input. You are tasked with preparing the following state,

$$\begin{aligned}|\psi(\alpha,\beta,\gamma)\rangle = &\cos\tfrac{\alpha}{2}\cos\tfrac{\beta}{2}\left[\cos\tfrac{\gamma}{2}|111000\rangle - \sin\tfrac{\gamma}{2}|000111\rangle\right] \\ &- \cos\tfrac{\alpha}{2}\sin\tfrac{\beta}{2}|001011\rangle - \sin\tfrac{\alpha}{2}|011001\rangle,\end{aligned}$$

where α, β, and γ are angles of rotation for `qml.SingleExcitation`, `qml.DoubleExcitation`, and your homemade triple-excitation operator. You also need to determine how to combine these three different Givens rotations in order to prepare $|\psi\rangle$. Specifically, your code must do the following:

- Provide a protocol for implementing a triple-excitation Givens rotation.
- Construct a quantum function that prepares the state $|\psi\rangle$ using some combination of one `qml.SingleExcitation` gate, one `qml.DoubleExcitation` gate, and one triple-excitation gate.

The template file `triple_givens_template.py` contains two functions. The `triple_excitation_matrix` function will construct the triple-excitation Givens rotation matrix, and the `circuit` function will prepare the state $|\psi\rangle$ and output `qml.probs`.

Input

- `list(float)`: A list containing the angles α, β, and γ in that order.

Output

- `list(float)`: The return of the `circuit` function: `qml.probs`.

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.

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

Version History

Version 1: Initial document.