QHack

Quantum Coding Challenges

Jump to code — Collapse text

8. Stuck in the Middle with U

0 points

Welcome to the QHack 2023 daily challenges! Every day for the next four days, you will receive two new challenges to complete. These challenges are worth no points — they are specifically designed to get your brain active and into the right mindset for the competition. You will also learn about various aspects of PennyLane that are essential to quantum computing, quantum machine learning, and quantum chemistry. Have fun!

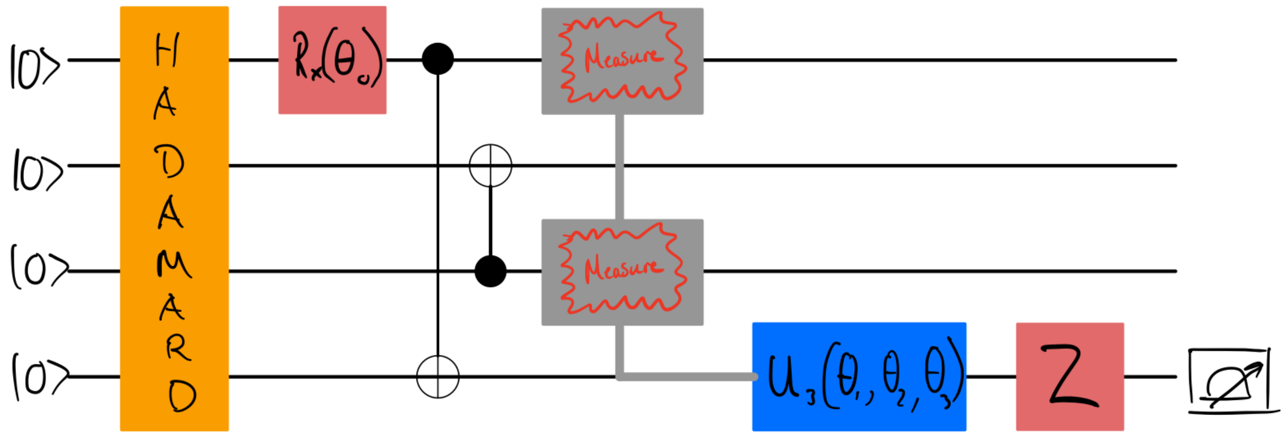
Tutorial #8 — Mid-circuit measurements

In classical computations, inserting control flow — e.g, `if` statements — right in the middle of a large computation is no problem at all since measuring variables does not affect the output of the computation. The same can't be said about quantum computations — if we measure, we better be careful!

In this challenge, you'll look at how mid-circuit measurements work in PennyLane.

Challenge code

In the code below, you are given a function called `circuit`. **You must complete this function** by constructing the following four-qubit circuit:



The circuit has a Hadamard gate on every qubit, an R_x gate, a couple of CNOTs, and then the mid-circuit measurements. Note here that the measurements happen on the first and third qubits, and that the `qml.U3` gate is only applied to the fourth qubit if the following condition is met upon measuring the first and third wires: $m_0 + m_2 \geq 1$ (i.e. at least one of them is 1). The last operator, `qml.PauliZ` on the fourth qubit, is applied regardless.

The `qml.measure` function should be helpful to you!

Input

As input to this problem, you are given:

- `angles` (`list(float)`): a list of angles containing θ_0, θ_1, θ_2, and θ_3 in that order. Use this to create the circuit!

Output

This code must output a `numpy.tensor` containing the probabilities associated to a computational basis measurement **on the fourth qubit.**

If your solution matches the correct one within the given tolerance specified in `check` (in this case it's a `1e-4` relative error tolerance), the output will be `"Correct!"` Otherwise, you will receive a `"Wrong answer"` prompt.

Good luck!

Code

Help

```python
import json
import pennylane as qml
import pennylane.numpy as np
```

```python
n_qubits = 4
dev = qml.device("default.qubit", wires=n_qubits)

@qml.qnode(dev)
def circuit(angles):
    """A quantum circuit made from the quantum function U.

    Args:
        angles (list(float)): A list of angles containing theta_
    Returns:
        (numpy.tensor): The probability of the fourth qubit.
    """

```

```python
    # Put your code here #

```

```python
    return qml.probs(wires=3)

```

```python
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:
    angles = json.loads(test_case_input)
    output = circuit(angles).tolist()
    return str(output)

def check(solution_output: str, expected_output: str) -> None:
    solution_output = json.loads(solution_output)
    expected_output = json.loads(expected_output)

    assert np.allclose(solution_output, expected_output, rtol=1e

```

```python
test_cases = [['[1.0, 1.5, 2.0, 2.5]', '[0.79967628, 0.200323
```

```python
for i, (input_, expected_output) in enumerate(test_cases):
    print(f"Running test case {i} with input '{input_}'...")

    try:
        output = run(input_)

    except Exception as exc:
        print(f"Runtime Error. {exc}")

    else:
        if message := check(output, expected_output):
            print(f"Wrong Answer. Have: '{output}'. Want: '{expe

        else:
            print("Correct!")
```

Copy all

Submit

Open Notebook

Reset