QHack

Quantum Coding Challenges

CHALLENGE COMPLETED

View successful submissions

Jump to code — Collapse text

Don't Hit the Ground

300 points

Backstory

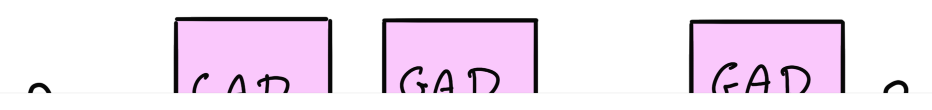
Zenda and Reece discuss strategies to interfere with the correct functioning of Sqynet, the conscious quantum computer that's taking over the galaxy. One way to tamper with its hardware is to bombard Sqynet's outer shell with plasma grenades, exposing the quantum computer to higher temperatures. As a consequence, Sqynet won't be able to prepare its ground state quickly.

Preparing ground states

Preparing a fiducial state, usually denoted by $|0\rangle$, is the first step before carrying out any quantum computations. For most quantum computers, this is a straightforward process (although sometimes energy and time consuming). We need to bring the quantum device to almost absolute zero so that it relaxes to its *ground state* —the state of minimal energy— which is our choice of fiducial state.

Why does this happen? Quantum systems are never really isolated, so they will exchange energy with their environment. The net effect is that any quantum properties of the system's state, i.e. superpositions and entanglement, are lost after some time.

How do we model this energy exchange at finite temperature? The *Generalized Amplitude Damping channel* provides a good approximation. Suppose γ is the photon loss rate at zero temperature, and p is the probability that a qubit emits a photon to the finite-temperature environment (the system will also absorb photons with probability $1-p$). We can approximate the interaction with the environment for a duration t via the circuit below.



That is, we compose many Generalized Amplitude Damping channels with infinitesimal noise parameters $\gamma\Delta t$ and de-excitation probability p. A shorter `step` Δt gives a more precise calculation, but we will need more Generalized Amplitude Damping channels to model the same duration T.

Zenda and Reece need to figure out a measure of how quickly Sqynet can relax its fiducial state, given some photon loss rate γ and emission probability p. Assuming that Sqynet is in the initial state

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

your task is to estimate the *relaxation half-life*, which is the time at which we obtain the outcome $|1\rangle$ with probability 1/4 (the measurement is performed in the computational basis).

Challenge code

You must complete the `half_life` function to calculate the time T at which the probability of measuring $|1\rangle$ becomes 1/4.

Input

As input to this problem, you are given:

- `gamma` (`float`): The zero-temperature photon loss rate.
- `p` (`float`): The de-excitation probability due to temperature effects

Output

This code will output a `float` equal to your estimate of the relaxation half-life. Note that you may require the step and iterations of your circuit to actually reach the half-life.

If your solution matches the correct one within the given tolerance specified in `check` (in this case it's an absolute tolerance of `0.2`), 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
def half_life(gamma, p):
    """Calculates the relaxation half-life of a quantum system that exchanges
    This process is modeled via Generalized Amplitude Damping.

    Args:
        gamma (float):
            The probability per unit time of the system losing a quantum of en
            to the environment.
        p (float): The de-excitation probability due to environmental effect

    Returns:
        (float): The relaxation haf-life of the system, as explained in the pr
    """

    num_wires = 1

    dev = qml.device("default.mixed", wires=num_wires)

    # Feel free to write helper functions or global variables here

    @qml.qnode(dev)
    def noise(
        gamma,  # add optional parameters, delete if you don't need any
    ):
        """Implement the sequence of Generalized Amplitude Damping channels in
        You may pass instead of return if you solved this problem analytically

        Args:
            gamma (float): The probability per unit time of the system losing
            to the environment.

        Returns:
            (float): The relaxation half-life.
        """
        # Don't forget to initialize the state
        # Put your code here #

        # Return something or pass if you solved this analytically!

    # Write any subroutines you may need to find the relaxation time here

    # Return the relaxation half-life

```

```python
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:

    ins = json.loads(test_case_input)
    output = half_life(*ins)

    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, atol=2e-1
    ), "The relaxation half-life is not quite right."
```

```python
test_cases = [['[0.1,0.92]', '9.05'], ['[0.2,0.83]', '7.09']]
```

```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: '{expected_output}'.

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

Copy all Reset Open Notebook Submit