

[SIGN OUT](#)

QHack

Quantum Coding Challenges

[RANK](#)[TEAM](#)[CHALLENGES](#)[SUBMISSIONS](#)[SUPPORT](#)**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.

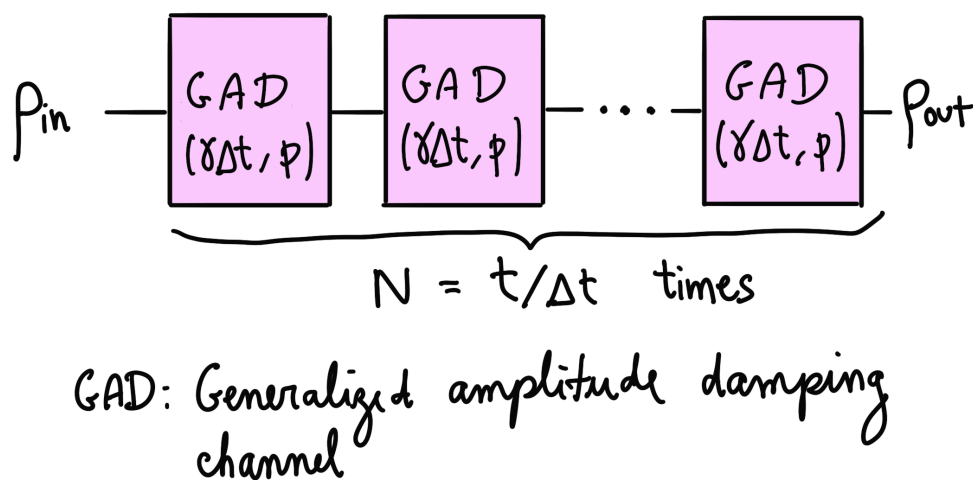
Preparing ground states

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



```
1 import json
2 import pennylane as qml
3 import pennylane.numpy as np
```



```

4 v def half_life(gamma, p):
5     """Calculates the relaxation half-life of a quantum system t
6     This process is modeled via Generalized Amplitude Damping.
7
8     Args:
9         gamma (float):
10             The probability per unit time of the system losing a
11             to the environment.
12         p (float): The de-excitation probability due to environm
13
14     Returns:
15         (float): The relaxation haf-life of the system, as expla
16     """
17
18     num_wires = 1
19
20     dev = qml.device("default.mixed", wires=num_wires)
21

```

```

22     # Feel free to write helper functions or global variables here
23
24     @qml.qnode(dev)
25     def noise(
26         gamma, # add optional parameters, delete if you don't n
27 v ):
28         """Implement the sequence of Generalized Amplitude Dampi
29         You may pass instead of return if you solved this proble
30
31         Args:
32             gamma (float): The probability per unit time of the
33             to the environment.
34
35         Returns:
36             (float): The relaxation half-life.
37         """
38         # Don't forget to initialize the state
39         # Put your code here #
40
41         # Return something or pass if you solved this analytical
42
43         # Write any subroutines you may need to find the relaxation
44
45         # Return the relaxation half-life
46

```

47 *# These functions are responsible for testing the solution.*

48 `def run(test_case_input: str) -> str:`

49

50 `ins = json.loads(test_case_input)`

51 `output = half_life(*ins)`

52

53 `return str(output)`

54

55 `def check(solution_output: str, expected_output: str) -> None:`

56 `solution_output = json.loads(solution_output)`

57 `expected_output = json.loads(expected_output)`

58 `assert np.allclose(`

59 `solution_output, expected_output, atol=2e-1`

60 `), "The relaxation half-life is not quite right."`

61

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

63 `for i, (input_, expected_output) in enumerate(test_cases):`

64 `print(f"Running test case {i} with input '{input_}'...")`

65

66 `try:`

67 `output = run(input_)`

68

69 `except Exception as exc:`

70 `print(f"Runtime Error. {exc}")`

71

72 `else:`

73 `if message := check(output, expected_output):`

74 `print(f"Wrong Answer. Have: '{output}'. Want: '{expe`

75

76 `else:`

77 `print("Correct!")`

 Copy all

Submit

Open Notebook 

Reset