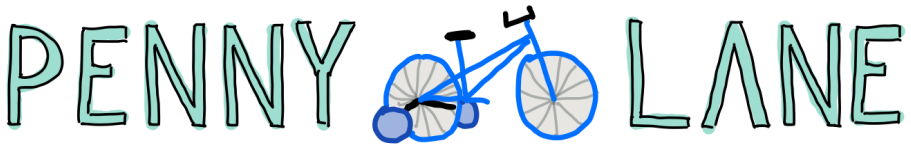

PennyLane 101: Superdense Coding [300 points]

Version: 1

PennyLane 101

The **PennyLane 101** challenges will introduce quantum computing concepts with PennyLane. Whether you're coming from an advanced quantum computing background, or you've never evaluated a quantum circuit before, these challenge questions will be a great start for you to learn how quantum computing works using PennyLane. Beyond these five questions in this category, there are well-developed demos and tutorials on the PennyLane website that are a good resource to fall back on if you are stuck. We also strongly recommend consulting the PennyLane documentation to see an exhaustive list of available gates and operations!

Problem statement [300 points]

Superdense coding is a communication protocol between two parties, which we call Alice and Bob. Using this algorithm, it is possible to transmit the two classical bits of information by transmitting only one qubit to the receiver. In the protocol, Alice and Bob share a maximally entangled pair of qubits A and B, represented by the state

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

where the first qubit A is initially in Alice's possession, and the second one B is in Bob's. Alice wants to send a number 0, 1, 2 or 3 in a message, which requires two classical bits. The protocol then works as follows. If Alice wants to send the number

- 0, then she does nothing to her qubit;
- 1, then she performs the Pauli X operation on her qubit;
- 2, then she performs the Pauli Z operation on her qubit;
- 3, then she performs the Pauli X operation followed by Pauli Z on her qubit.

After manipulating it in this manner, Alice sends her qubit to Bob. Now that Bob has the pair of qubits, he performs a CNOT operation with his newly received

qubit A as the control. Next, he acts on A via a Hadamard gate and proceeds to measure the two-qubit system in the computational basis. A straightforward calculation shows that Bob's post-measurement state is uniquely-determined by the number Alice sent. If Alice sent the number 0, then Bob's post-measurement state will be $|0\rangle_A |0\rangle_B$; if she meant to send the number 1, then Bob's post-measurement state will be $|0\rangle_A |1\rangle_B$; and so on. In this way, Bob can guess the number that Alice sent him with 100% accuracy.

This procedure is dependent on the ability to generate maximally entangled states. Suppose that we have an imperfect entangler that generates the following state instead:

$$|\psi\rangle = \cos(\alpha) |0\rangle_A |0\rangle_B + \sin(\alpha) |1\rangle_A |1\rangle_B .$$

Your code will implement this protocol to show that, for $\alpha \neq \frac{\pi}{4}$, Bob can only guess Alice's original bits correctly with some probability (you must calculate this).

The provided template `superdense_coding_template.py` contains a function called `superdense_coding` that you must complete. In this function, you will code Alice's and Bob's protocols and output the probability that Bob reads out the correct state. For convention in your code, please use `wires=0` as Alice's qubit and `wires=1` as Bob's qubit.

Input

- `list`: A list containing the angle α that defines $|\psi\rangle$ and the integer value (0, 1, 2, or 3) that Alice wants to send to Bob.

Output

- `float`: The probability of Bob guessing Alice's number that she sent.

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.