Quantum Games: The CHSH game [200 points]

Version: 2

Games

Quantum computers help us solve many problems, but we can also use them to explore some of the most mysterious aspects of quantum mechanics. We learn so much better when we have fun, so why not use some crazy games and experiments to explore the boundaries of quantum theory? These fun coding challenges, ranging from entangling full-blown animals to using quantum circuits to cheat our way into victory, teach us a lot about why quantum computing is so powerful.

In the **Games** category, we will be exploring some of the weirdest quantum experiments proposed in the literature. These will enlighten us about how quantum mechanics is different from classical physics, but will also give rise to deeper philosophical questions. Let's have some fun!

Problem statement [200 points]

The CHSH game provides an example of a quantum strategy that increases the probability of winning over any possible classical protocol. The game is collaborative between two parties, Alice and Bob, who are given two random bits x and y, where x is known by Alice and y is known by Bob. Alice and Bob will additionally select two values, a and b, which can be 0 or 1, and will win if

$$x \cdot y = a \oplus b,$$

where $\oplus$ represents addition modulo 2. Since the probability of $x \cdot y = 0$ is 75%, the best classical strategy is that both of them previously agree on choosing either $a, b = 0, 0$ or $1, 1$, ignoring the bits they received.

Can Alice and Bob do better if we provide them with an entangled pair of qubits? Let's consider the following quantum strategy. Alice and Bob are provided with an entangled pair of the form

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|11\rangle\,,$$

which is neither normalized nor maximally entangled. For simplicity, let us keep the weights α and β real. With this state given to them, Alice and Bob devise a way to win the game by making conditional measurements in different bases.

Alice and Bob *separately* choose a measurement basis $\{|\nu_0(\theta)\rangle\,, |\nu_1(\theta)\rangle\}$, where

$$|\nu_0(\theta)\rangle = \cos(\theta)\,|0\rangle + \sin(\theta)\,|1\rangle$$

and

$$|\nu_1(\theta)\rangle = -\sin(\theta)\,|0\rangle + \cos(\theta)\,|1\rangle\,.$$

For her basis, Alice chooses $\theta = \theta_{A0}$ if she receives $x = 0$, and $\theta = \theta_{A1}$ if she receives $x = 1$. Similarly, Bob will choose $\theta = \theta_{B0}$ if $y = 0$ and $\theta = \theta_{B1}$ if $y = 1$. Having chosen their angles that define their respective measurement bases, they measure $|\psi\rangle$! If Alice's measurement result corresponds to ν_0 (ν_1), she chooses $a = 0$ (1). Bob's choice is made in a similar fashion.

Your task is to build the quantum circuit that implements the measurements in the bases above and provide a set of angles that Alice and Bob should choose to maximize their probability of winning. Your output should be the maximal probability given α and β.

The provided template `CHSH_game_template.py` contains a few functions that you need to complete:

- `prepare_entangled`: a protocol for preparing the entangled state $|\psi\rangle$.
- `chsh_circuit`: constructs a circuit that implements Alice's and Bob's measurements in the rotated bases $\{|\nu_0(\theta)\rangle\,, |\nu_1(\theta)\rangle\}$.
- `winning_prob`: returns the probability of Alice and Bob winning the game.
- `optimize`: optimizes the angles $\theta_{A0}, \theta_{A1}, \theta_{B0}, \theta_{B1}$ to maximize the probability of Alice and Bob winning the game.
- `cost`: defines the cost function used in `optimize`.

Input

- `list(float)`: A list containing the unnormalized coefficients of the entangled state: α, β.

Output

- `float`: The probability of Alice and Bob winning the game when they use the optimal quantum strategy

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 solution 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. Version 2: Added `cost` function in the list of functions that the user needs to define. This update does not affect submissions that were evaluated as correct before this change.