Quantum Games: Elitzur-Vaidman Bomb [300 points]

Version: 1

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 [300 points]

The Elitzur-Vaidman bomb tester showcases one of the non-intuitive features of quantum mechanics that come from superposition. This mechanism is a quantum circuit that can test whether a bomb is live or a dud without necessarily triggering it. The circuit is shown in the form of an interferometer in Figure 1.

The light-blue boxes in Figure 1 denote beam splitters, which are optical devices that have two entrance and two exit ports for photons. The state of an incoming photon is denoted by $|0\rangle$ or $|1\rangle$, depending on the entrance port. We use a similar notation for the states of the outgoing photons. The convention for the ports used throughout this challenge is shown in Figure 2.

The action of the beam splitter $BS(\theta)$ is represented by the unitary operator

$$U_{BS}(\theta) = R_y(2\theta).$$

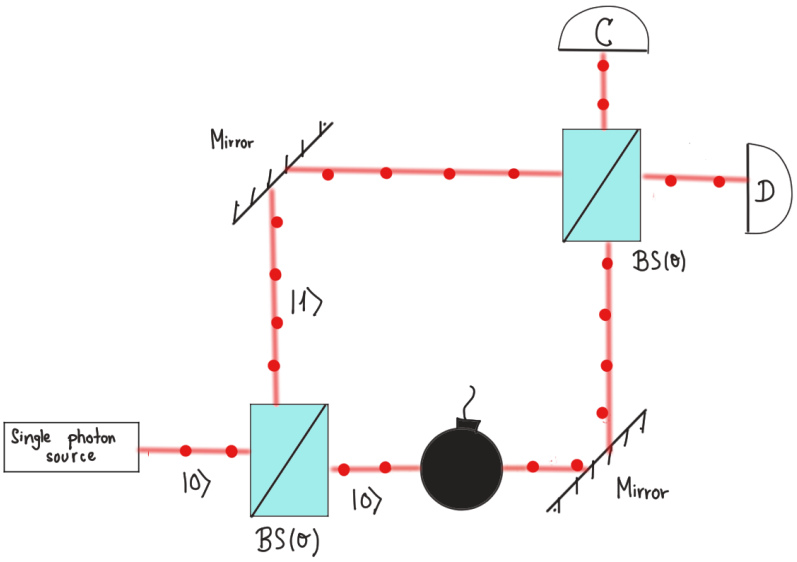

Figure 1: Interferometer bomb tester

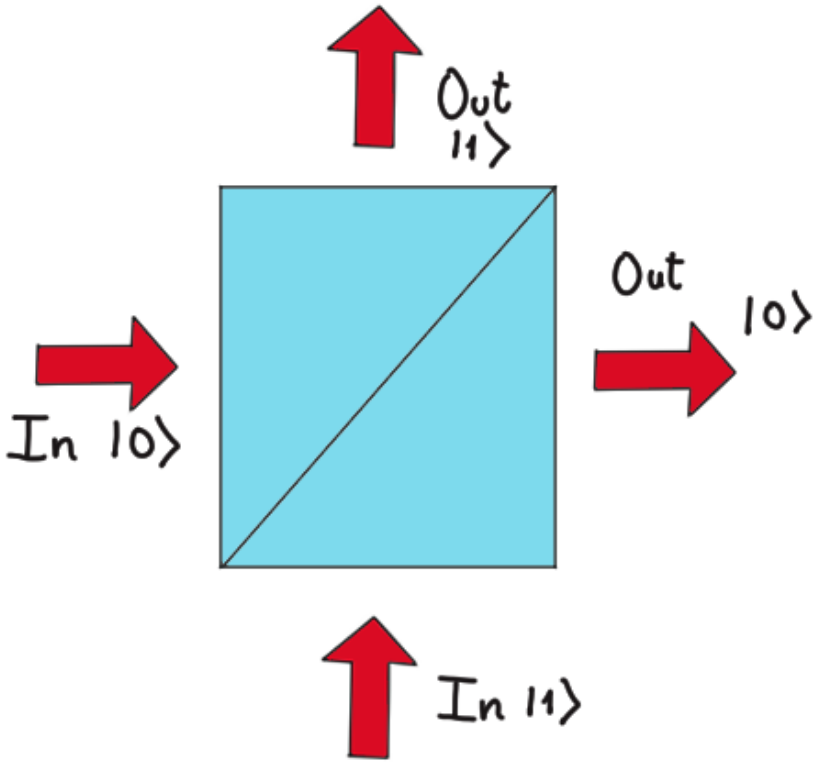

Figure 2: Beam splitter port convention

As shown in Figure 1, a bomb that explodes upon contact with a single photon is placed in the branch corresponding to the output $|0\rangle$, which we call branch 0. If this bomb is live, it will explode with a probability equal to the probability that the photon goes through that branch. Therefore, the exploding bomb is de facto a measurement of the photon's position. If the bomb is a dud, it acts as a completely transparent material and does not affect the photon (it is the identity operator). At the end of the circuit, there are two detectors C and D.

Interestingly, when there is no functional bomb present, only detector C will beep. However, if there is a genuine bomb, and if the bomb does not explode, either detector may beep. Thus, if detector D beeps at all, we have detected a real bomb without exploding it! But if detector C beeps, we cannot say anything about the bomb. Feel free to calculate these results by hand to understand the setup better!

We would like to find the number of bombs that do not explode and that we can certify as live. This number is increased or reduced by adjusting the value of θ or by concatenating the circuit many times, as shown in Figure 3.

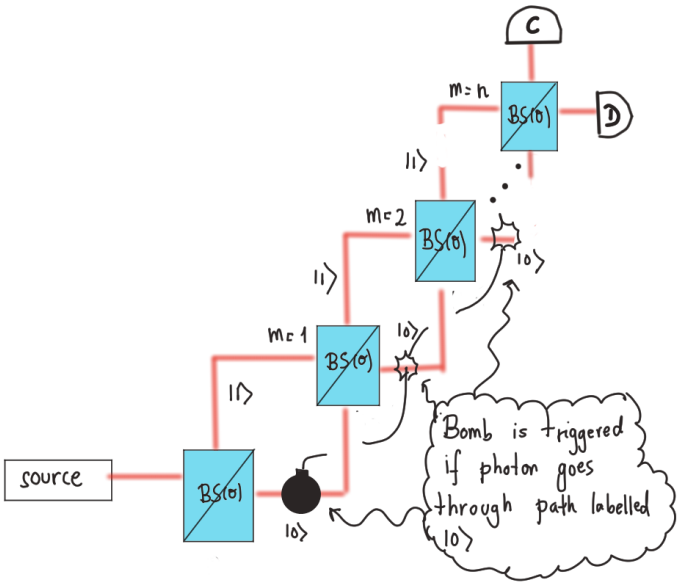

Figure 3: Iterative bomb test

Your task is to write a circuit that concatenates n interferometers with detectors C and D at the end of the chain. For the purposes of this problem, we assume

that the bomb is always live and that a photon going through branch 0 in any of the interferometers in the chain will trigger the bomb (the trigger of the bomb passes through all the interferometers, as shown in Figure 3).

Your code will keep count of the possible results (explosion, detection at C, and detection at D) of 10000 one-shot measurements. As the output of your program, you should calculate the probability p of detector "D" beeping, given that the bomb did not explode:

$$p = \frac{\text{Number of times that D beeps}}{\text{Number of bombs that did not explode}}.$$

The provided template `Elitzur_Vaidman_template.py` has three functions that you need to complete:

- `is_bomb`: generate a sample measurement of size 1 as performed by the bomb, after the photon passes through **one** beam splitter
- `bomb_tester`: implement the beam splitter right after the bomb and the measurement in the computational basis performed by the detectors. It should also return a sample of size 1.
- `simulate`: concatenate `n` bomb circuits and a final measurement, collect the results of 10000 one-shot measurements that correspond to the number of times that D beeps and the number of times that an explosion occurred, and calculate p.

Note: Although technically the mirrors in the interferometers would need to be introduced in the circuit as well, they have no effect on the probabilities calculated here.

Input

- `list`: A list containing the angle θ characterizing the beam splitter and the number of times n that the circuit is iterated.

Output

- `float`: probability of testing a true bomb given that it didn't explode

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 `0.05` 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 in the template 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: **0.05**
Time limit: **60 s**

Version History

Version 1: Initial document.