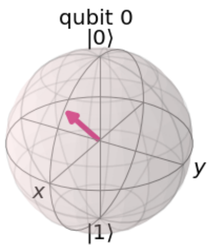
Coding assignment write-up

Mohammad Sheikhattari

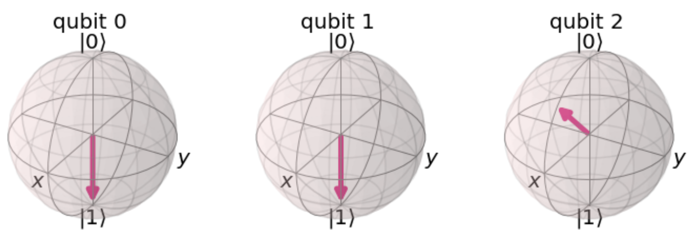
April 15, 2022

1 Teleportation

The code for this problem was designed by following exactly the teleportation algorithm discussed in class, and the program was written using Qiskit. I have followed the exact same procedure as we did in class, consulting the Qiskit documentation for necessary methods such as conditional gate operations, or measurements, for instance. The code was tested for correctness by attempting to teleport an arbitrary state and evaluating whether the state was teleported correctly using a visualization of the input and output states. A random statevector was created and plotted on the bloch sphere, and at the end of the code the states of the qubits in the circuit were plotted to evaluate whether this random statevector was teleported. The visualizations are saved to output files, and several iterations of the code proved that the code works as intended, as each time the input state matched the state that was left in the third qubit of the circuit, indicating the state was indeed successfully teleported. The code can be tested simply by running it and verifying that the ”input_state.png” matches the state of the third qubit of ”output_state.png”. As a demonstration, one such instance of running the code is left here, as both images are included from that run - the states are the same. A circuit has also been drawn to further show that this is the same procedure as was taught in class.



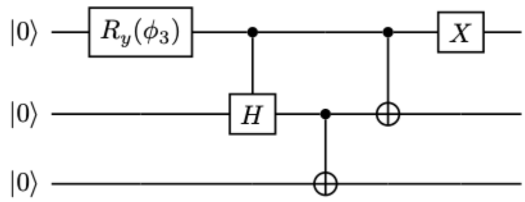
Input state:



Output state (qubit 2):

2 Distinguish states

The code for this problem was designed by recognizing the input states as being very similar to a well known state, the W state. W is the state of three qubits: $\frac{1}{\sqrt{3}}(|001>+|010>+|001>)$. The circuit for producing the state is well-known and shown below (https://en.wikipedia.org/wiki/W_state):



Getting to the desired states is as arbitrary as adding a phase to the second and third qubit using an Rz gate. So, now that we know how the states can be produced, we can easily solve the problem as follows. Take in one of the two states as input, and for the circuit, have the inverse of the circuit that takes you from the state $|000>$ to either one of these states. This will turn that state into $|000>$ since we applied the inverse of the circuit that takes you from $|000>$ to this state. It is arbitrary which we choose since they are orthogonal, and any unitary operation will preserve orthogonality, therefore if we transform to state $|000>$ for one state the other must be in a different state. Let's choose state $|\psi_0>$. We use the same circuit as shown above, with an added Rz gate of phase $\frac{2\pi}{3}$ on the second qubit and an Rz gate of phase $\frac{4\pi}{3}$ on the third qubit. The inverse of this circuit, applied to state $|\psi_0>$ will result in state $|000>$, while applied to state $|\psi_1>$ will result in some other state. A measurement on the resulting state, and observing which outcome occured tells us whether we were given state $|\psi_0>$ or $|\psi_1>$ and we can produce a corresponding output indicating so. The state $|\psi_0>$ results only in outcome $|000>$ with p = 1, while $|\psi_1>$ results in a superposition of states, so we can say we are in $|\psi_0>$ if we observe only a single measurement outcome over 1000 simulations, otherwise we are in $|\psi_1>$.

The code was tested by preparing and passing in $|\psi_0>$ and $|\psi_1>$ and confirming that the output corresponded to which state was passed in - this was indeed the case. Further, a loop was run 100 times that prints "FAIL" if in any iteration the outcome of distinguishing $|\psi_0>$ or $|\psi_1>$ did not agree with the expectation. Not once was "FAIL" printed, strongly supporting the success of the algorithm.