Bloch sphere (8 min)

The qubit, or quantum bit, is the quantum analogue to the classical unit of information (bit). Nevertheless, there are fundamental differences. Rather than having only two possible classical states (0 and 1), the qubit can be in a complex superposition of both states.

The qubit can be represented as a column vector. Alpha and beta are probability amplitudes, and their squared values, alpha squared and beta squared, are interpreted as the probability of measuring the associated outcome.

Since the probabilities of all possible outcomes of a given event must sum to one, we can establish that |alpha squared| + |beta squared| = 1.

This normalization rule allows for an alternative expression of a qubit state.

The canonical visualization for a qubit state is a point on the surface of a Bloch sphere. This is convenient for representing qubits and their operators because of the isomorphism between SU(2) (Special unitary group of degree 2, which contains all single-qubit unitary operations) and SO(3) (the group of all rotations about the origin of three-dimensional Euclidean space). (The global phase of the qubit is ignored, since it would be irrelevant anyway).

With the Bloch sphere, we have the fundamental basis states at the poles along the Z axis, with complex superpositions represented at any point in the surface between them. When a qubit is measured, it always collapses to one of these poles (basis states), and it stays in the exact same states throughout subsequent measurements.

However, the description of a quantum system is more than the sum of the descriptions of its individual qubits. This is due to entanglement, and it implicates that multi-qubit entangled states cannot be represented by simply adding a Bloch sphere for each extra qubit.

Superposition and entanglement are the two fundamental properties of quantum systems that make it possible for quantum computers to unlock algorithms more powerful than what is possible on quantum computers.

Gates and basis (15 min)

A quantum gate may be represented as a matrix, since a quantum operation is expressed as the product between the matrix and the vector representing the quantum state. The vector representing an n-qubit quantum state has length 2^n, and an n-qubit quantum gate is expressed by a 2^n times 2^n matrix.

Notice how the size of the quantum state vector, as well as the quantum operation, grow exponentially (powers of 2) with the number of qubits in the system, and you may begin to understand why simulating quantum systems, or quantum computers, on classical computers is such a hard task. The full state vector of a 50 qubit system would have 1x10^15 entries (of course, there are ways to optimize simulations at such a scale by making a few assumptions).

So let's start building a circuit! We start by importing the relevant python libraries, including qiskit modules. Here is a function I have built do execute a circuit and show execution results (we'll get back to it later).

We start by defining a quantum register of 2 qubits, and a classical register of two bits (one for each qubit we want to measure).

The physically implementable set of gets supported in current IBM quantum devices is the identity, a set of unitary gates which describe any possible unitary operation on a single qubit, and the CNOT, a controlled-not operation. The software however, handles a much larger set of gates which is always decomposed into these 5. Let's go over the most important ones:

X

Z

H

For a single qubit, any two orthogonal quantum states work as a basis. For convenience, we call the base we measure our qubits on, the computational basis. 0 and 1 is just a convenient choice of naming since it relates well to the bit in classical computing (we could very well name them anything else). In multi-qubit systems, we take the computational basis for each qubit, and then apply the tensor product between all these bases - the result is still a full set of orthogonal states. On the other hand, we can define the superposition basis, which for a single qubit, is the equal superposition of 0 and 1.

The Hadamard gate can be used to interchange the state of the qubit between these bases. Example: if we measure the |+> state on the computational basis, we will know that the state is in either + or -, but not which one, since the only values we can estimate are alpha and beta squared (which are always non-negative, independently of the non-squared probability amplitude). By applying an hadamard before measurement, we are in practice measuring a qubit in the superposition basis (since if we measure 1 we know the state was -, and if we measure 0 we know the state was +).

To have a universal quantum circuit, we need multi-qubit operations. The CNOT, also known as controlled-X, is the fundamental two-qubit gate; together with a generalized single qubit unitary, we are able to decompose any multi-qubit operation.

So let's see the CNOT in action.

We just created a superposition of entangled states!

Deutsch-Josza Algorithm (15 min)

Let's implement a specific, easy to understand algorithm to demonstrate not only a case of quantum speedup, but also how one goes about implementing arbitrary quantum circuits in Qiskit.

Here's the problem we want to solve - we have a function that maps an n bit string, to a bit output (either 0 or 1). We don't know the logic, or the inner workings of the function. However, we are promised that it is either constant or balanced.

We are given a black box, also known as oracle, that for each possible input, returns the function's output.

A classical approach to the circuit would be to call the function over each argument until we have enough data to provide an answer - in the 1 bit case, we always need two function evocations.

In a quantum setting, our black box need to allow for reversible computation, so we designe it as follows.

From this, the algorithm's circuit can be generalized to an n-bit input function-

Deutsch-Josza's algorithm illustrates the notion of **quantum parallelism**: a quantum register has the ability to exist in a superposition of base states - each one may be thought of as a single argument to a function.

A function performed on the register in a superposition of states is thus performed on each of the components of the superposition, _while only being applied once_.