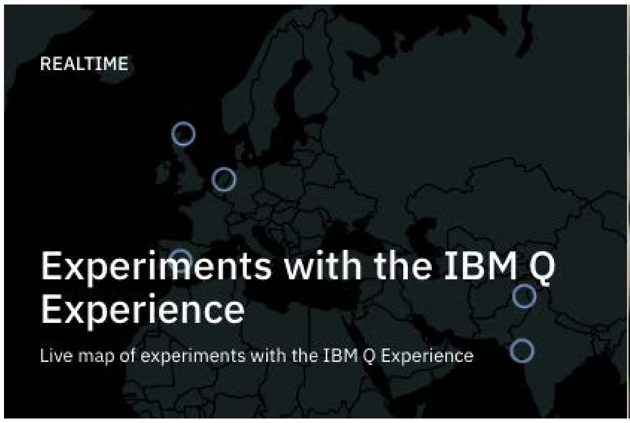
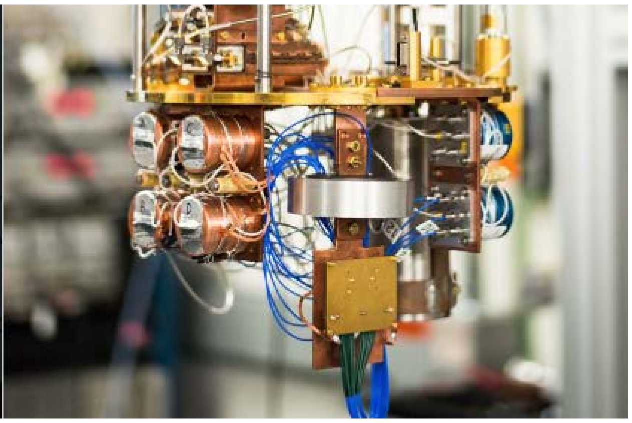

REALTIME
Experiments with the IBM Q Experience
Live map of experiments with the IBM Q Experience


Introduction

The IBM Q Experience platform allows anyone with access to the internet to write and run quantum algorithms on a real quantum computer. In this section, you will receive the necessary resources to be able to implement algorithms on the IBM platform, but we encourage you to register on the IBM page to access the composer's graphical interface (alternatively, if you are comfortable writing text-based programs, the interface here may be just fine).

The IBM Q composer allows one to express quantum circuits and quantum algorithms in a simple graphical way, and on the IBM QE site you will be able to save the results of your programs and access them later. To introduce you to the use of the platform IBM has a user beginner's guide. If you want to learn how to implement more complicated algorithms, we recommend you visit their full user guide. For researchers, it is now possible to request exclusive access.

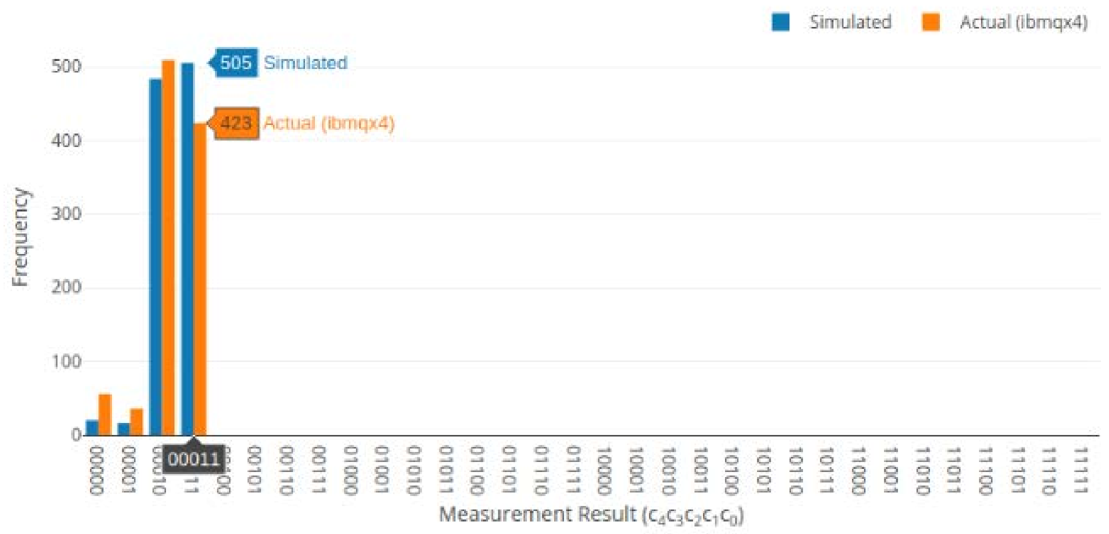
Once in the composer, you can create a new experiment and choose if you want to run your program in a real quantum computer or to simulate it. If you decide to run your code, you will have to choose the number of times that you want to run the code and which backend you will use. There are four possible backends in which you can run your codes. In this course, you will only use either the numerical simulation, or one of the five-qubit backends, ibmqx2 (may be unavailable due to maintenance) and ibmqx4 (when available). You will learn more about the backend options in the following sections.

The IBM Q Experience platform not only allows you to use the composer, but it also allows you write your code using Open QASM (Quantum Assembly Language). This is a simple text language that has elements of C and assembly languages. By using QASM you can describe any quantum circuit, in principle, since it has built in a universal set of quantum gates. In this section, you will learn how to express quantum circuits and write quantum algorithms by using QASM. If you want to learn more about the theory behind QASM we recommend you to read the following paper written by IBM researchers (including former MIT graduate students): Open Quantum Assembly Language.

What are the differences between going directly to IBM Q Experience platform, and using the interface provided directly within this course? Here, in this course, you will receive detailed instructions on how to use QASM, we will guide you step by step on how to create your first programs, and we will teach you how to read your results. In this first set of exercises, you will learn about syntax and common mistakes, measurements, single-qubit gates, two-qubit gates, and backends. You will have the opportunity to practice your knowledge and to prove your skills in the end-of-week test. Besides the results that the IBM platform gives you, you will also make a theoretical analysis of the quantum state evolution, so you will easily see the correspondence between the analytical and experimental results.

The numerical simulation employed here is based on the IBM QISKit engine, an open-source package which allows inclusion of gate and measurement noise, for added realism. This noise model is based on recent calibration data from the 5-qubit IBM QE machines, and the output histograms are equivalent to that you would see from real quantum computers.

Each exercise begins by default by employing the numerical simulation, to give you the most rapid feedback for your answer submission. With select exercises, once your submission is correct, according to the automated grader, your QASM program will automatically the be submitted for execution on a real quantum computer. Once it's run, you should see a plot of the results, comparing the simulated and real quantum computer's results, side-by-side, like this:



You'll need to be patient, however, because these run requests must be queued for execution, since hardware availability is limited, and each program must be run multiple times, to provide output statistics needed for pedagogical explanation.