QCHacks 2021 - Q-CTRL Challenge

Team: Fizziqs

To implement a high fidelity NOT gate and Hadamard gate on the Q-CTRL qubit, we focused on three aspects: calibration of the qubit to determine the Rabi rate, smoothing the input pulse, and optimizing the controls through a closed-loop optimizer.

1. Calibration/Determination of Rabi Rate

To determine the Rabi rate, which maps the hardware amplitudes to the control pulse amplitudes, we performed a Rabi experiment for multiple input amplitudes (between 0 and 1). This was done with the help of the Opal Boulder tutorial on pulse calibration. For example, for the input amplitude 0.7, we sent varying durations of a constant rectangular pulse to the QCHacks cloud qubit and then recorded the qubit population. See figure 1. By fitting to the following cosine function:

$$A\cos(2\pi\Omega t + \phi)^2$$

we can determine the Rabi rate for the associated input amplitude.

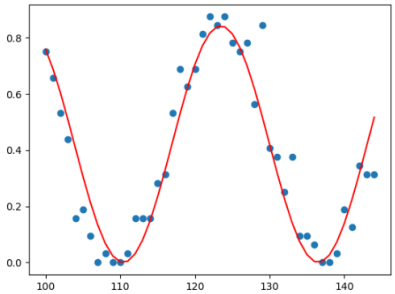

Figure 1: Qubit population over time (ns) for input amplitude 0.7.

After repeating for multiple input amplitudes, we find a linear relationship between the input amplitude and the Rabi rate.

$$rabi\ rate = 0.3008 * input\ amplitude + 0.0163$$

This allows us to create an initial pulse to send to the qubit from which we can then optimize. The initial pulse was determined using the rabi rate found as well as the Opal Boulder section on creating ideal plot pulses.

Ω_{NOT}:0.3171, duration=9.9076

Ω_{H}:-0.3171j, duration=4.9538

0.3171+0j, duration=4.9538

0.3171+0j, duration=4.9538

2. Pulse Smoothing

To smooth the input pulse, we wrote a filter function that discretizes the input signal into smaller sample steps. It then performs a convolution of the discretized input pulse with the Hann

window function of the time period of the pulse in order to create a smoother pulse curve and prevent large discontinuous jumps of the signal. We then apply a Planck-taper window filter to the pulse to smoothly bring the two endpoints to 0. This allows us to append signals to each other to create a longer signal that is still continuous.

3. Controller optimization.

We chose to implement the closed-loop automated optimization approach because of the overall lack of understanding of the qubit system, where in particular, it seemed difficult to create an error term for the state preparation and measurement (SPAM) errors. We used Q-CTRL's closed-loop optimizer using a series of test controls to randomly generate target states which we can use to evaluate with the measurements. We decided to optimize both types of gates simultaneously, since we wanted to keep the phase between them constant and ensure that cascading the gates works as expected.
We chose 5 input controls: the first two are the test NOT and Hadamard gates, and the last three are pulses of *n* sequential gates, where *n* is randomly chosen between 2 and 7.

Control pulse	**Example: n = 5**
1. A single NOT gate	N
2. A single Hadamard gate	H
3. A series of *n* NOT gates	N N N N N
4. A series of *n* Hadamard gates	H H H H H
5. A series of *n* NOT and Hadamard gates, ordered randomly	N H H N H

Table 1: Test suite of control pulses used in optimizer to generate infidelity values of NOT and Hadamard gates.

The target states are calculated by applying the actual NOT and Hadamard gates (where $\Phi_1 = 0$, $\Phi_2 = 0$, $\theta = 0$) in the chosen orders on a zero-qubit. The error is then the absolute value difference between the probability of one in the measurement and the probability of one in the final ideal state, divided by *n* to normalize the error. From the entire test suite, the overall infidelity value of the NOT and Hadamard gates is the average of all errors from the input control pulses.

To optimize our controls, we wrote a main script, which sends a control pulse or series of control pulses to the cloud qubit and records the results with the least amount of error. It repeats this, using the automated closed-loop optimizer from Q-CTRL to generate the next set of test points, until the error is less than a threshold value.