# RL-QOC report

Ding Ruiqi

September 2020

## 1 Introduction

The objective of this experiment is to use Deep Reinforcement Learning (DRL) to generate control signal that drives one quantum state to another.

The motivation is to utilize the aggressive behavior of RL, which means that the RL agent will try to maximize the reward function at each time step, this is equivalent to trying to achieve the goal with minimum number of time steps. In comparison, for currently used algorithm known as GRAPE (GRadient Ascent Pulse Engineering), the total number of time steps must be fixed during the gradient ascent optimization process.

## 2 Background

### 2.1 Reinforcement Learning

In RL, there is an agent and an environment. The agent observes the environment state, and takes an action based on some policy. This action drives the environment to the next state, and a corresponding reward is given to the agent. The goal of RL is to find the best policy that produces maximum amount of total reward.

### 2.2 Deep Reinforcement Learning

RL has a long history before the appearance of artificial neural network. It has been studied in the mathematical branch Dynamic programming for a

long time. In DRL, the only difference from RL is that the policy is parameterized by a deep Neural Network (NN) denoted by $\theta$. Because mathematically a policy is a stochastic function that maps an observation to an action:

$$\underbrace{a_t}_{\text{agent action}} \sim \underbrace{\pi_{\underbrace{\theta}_{\text{NN}}}}_{\text{stochastic policy parameterized by NN}}(\cdot|\underbrace{s_t}_{\text{game state}}) \tag{1}$$

Before NN, the policies are usually completely stored as numerical values, which means to store the best action for each possible observation. This is huge amount of data and the reason why people use NN to approximate the policy. Because from a mathematical point of view, NN is essentially the list of coefficients of a highly nonlinear function, which can be optimized easily to approximate the target function that requires much more numbers to be represented.

## 2.3 The environment

Mathematically, the environment is a simple stochastic function (usually natural law) that maps the state and action from a time step to the next time step. Note that this immediately indicates the Markovian property of the agent-environment interaction process:

$$s_{t+1} \sim \underbrace{P}_{\text{natural law}}(\cdot|s_t, a_t) \tag{2}$$

## 2.4 The entropy-regularized return

In RL terminology, *return* is defined as the weighted summed of the *reward* from each time step. And at each time step, apart from the reward $r_t$ given by the environment, we also add the entropy (defined as $H(P) = \underset{x\sim P}{E}[-\log P(x)]$) of the policy. This encourages the agent to explore various possible actions and prevent it from getting stuck at some local maximum, because a higher entropy means distributing probabilities among more actions.

$$R_\tau = \sum_{t=0}^{\infty} ( \underbrace{\gamma}_{\text{discount}} )^t (r_t + \alpha \underbrace{H(\pi(\cdot|s_t))}_{\text{entropy of the policy}} ) \tag{3}$$

As the above equations and explanations are sufficient for the main discussions, we will not discuss the use of Bellman equation and the Soft Actor-Critic (SAC) algorithm in finding the best policy.

# 3 The experiment

## 3.1 The workflow

In the experiment we use the SAC algorithm from a package called *stable-baselines3* developed by German Aerospace Center Institute of Robotics and Mechatronics. And we write the environment using the standard provided by Elon Musk's company OpenAI, i.e., write a environment class that inherits from the OpenAI's *gym.Env* class. Specifically, we will define the following:

- Action space and observation space
- System Hamiltonian and initial state
- A function called *step* that involves the following:
  - The evolution of the system
  - The reward and cost function
  - The termination criterion

## 3.2 Action space and observation space

Let $\psi$ be the state of the quantum system. Let $f$ be the fidelity between $\psi$ and the target state $|1\rangle$. Let $a$ be the action of the agent, which can be either the control amplitude $u$ or the gradient of it. Then the observation $o$ is chosen to be

$$o = [\mathrm{Re}(\psi), \mathrm{Im}(\psi), f, a] \tag{4}$$

When $a$ is the control amplitude, we have

$$a^{(1)} = u \tag{5}$$
$$H(t) = H_0 + a_0(t)\ \hat{x} + a_1(t)\ \hat{y} \tag{6}$$

When $a$ is the gradient of the control amplitude we have

$$a^{(2)} = \frac{du}{dt} \tag{7}$$
$$H(t) = H_0 + \text{Clip}[\int a_0(t)dt]\ \hat{x} + \text{Clip}[\int a_1(t)dt]\ \hat{y} \tag{8}$$

The motivation to use $a^{(2)}$ is to limit the gradient of the control amplitude $u$ so that the control signals are smooth.

### 3.3 System Hamiltonian and initial state

The goal of the agent is to drive an initial quantum state to a final quantum state through a series of action, we call this an *episode*. When an episode is finished, the environment is reset with a new randomly chosen initial state. Note that the target state should not change. In this way the agent will learn to reach a target starting from different initial states. I have tried to randomly reset both the initial and target states, but the agent was unable to learn at all.

Two different systems have been tested. The first case is a simple quibit. Let $b$ be the annihilation operator for a qubit:

$$\text{sys}_1 : \begin{cases} H_0 = b^\dagger b \\ \hat{x} = b^\dagger + b \\ \hat{y} = i(b^\dagger - b) \end{cases} \tag{9}$$

The second case is a 3-level approximation to a Transmon with parameter adopted from the DRAG paper:

$$\text{sys}_2 : \begin{cases} H_0 = \sum_{j=1}^{2} \delta_j |j\rangle\langle j| \\ \hat{x} = \sum_{j=1}^{2} \frac{\lambda_j}{2} (|j\rangle\langle j-1| + |j-1\rangle\langle j|) \\ \hat{y} = \sum_{j=1}^{2} \frac{\lambda_j}{2} i(|j\rangle\langle j-1| - |j-1\rangle\langle j|) \end{cases} \tag{10}$$

with the following parameters

$$\text{anharmonicity: } \Delta = -0.4 \quad \text{GHz} \tag{11}$$
$$\delta = [0, \Delta] \tag{12}$$
$$\lambda = [1, \sqrt{2}] \tag{13}$$

## 3.4 The evolution of the system

The evolution of the system is calculated using the *qutip* function *mesolve*.

## 3.5 The reward and cost function

Several cost functions has been tested:

$$c_1 = 0 \tag{14}$$
$$c_2 = \Theta(f - 0.9)\langle|u|\rangle \tag{15}$$
$$c_3 = (f - \frac{f_0 + 1}{2})^2 \langle|u|\rangle \tag{16}$$

where $\Theta()$ is the Heaviside step function, and $f_0$ is the fidelity of the initial state.

The use of $c_1$ is to verify that without some cost function constraint, the agent can successfully learn the task.

The use of $c_2$ is motivated by the physical consideration that we want the amplitude to drop to zero when the desired fidelity is achieved. By using the Heaviside step function, when the fidelity is still small, we don't penalize the agent for producing high amplitude, once the fidelity reached a threshold of

0.9, which means the episode is about to end, we penalize the agent by the mean absolute value of the control amplitude.

The use of $c_3$ is motivated by the physical consideration that we want the amplitude to be small both at the beginning and the end of the episode. Therefore we use a quadratic function centered at the fidelity midpoint. Thus at the beginning of the episode, there is a large weight for the amplitude cost. In the middle of the episode, the weight for amplitude cost is small, and at the end of the episode, the weight is large again.

The reward function is chosen to be the fidelity increase with cost function subtracted from it:

$$r = f_t - f_{t-1} - c \tag{17}$$

## 3.6 The termination criterion

The episode terminates when the first or both conditions below are satisfied:

$$\text{cd}_1: \quad f > f_{\text{targ}} \tag{18}$$

$$\text{cd}_2: \quad \langle |u| \rangle < u_{\text{term}} \tag{19}$$

## 3.7 The result

In the following table I show the experiment results for different combination of **systems, actions, cost functions and termination conditions**. The following abbreviation is used:

- S: The agent successfully learns.
- F: The agent fails to learn.

| case | system | action | cost | termination condition | result |
|---|---|---|---|---|---|
| 1 | $\text{sys}_1$ | $a^{(1)}$ | $\text{c}_1$ | $\text{cd}_1$ | S |
| 2 | $\text{sys}_1$ | $a^{(2)}$ | $\text{c}_1$ | $\text{cd}_1$ | S |
| 3 | $\text{sys}_1$ | $a^{(2)}$ | $\text{c}_2$ | $\text{cd}_1$ | S |
| 4 | $\text{sys}_1$ | $a^{(2)}$ | $\text{c}_2$ | $\text{cd}_1$ and $\text{cd}_2$ | S |
| 5 | $\text{sys}_2$ | $a^{(1)}$ | $\text{c}_1$ | $\text{cd}_1$ | S |
| 6 | $\text{sys}_2$ | $a^{(2)}$ | $\text{c}_1$ | $\text{cd}_1$ | S |
| 7 | $\text{sys}_2$ | $a^{(2)}$ | $\text{c}_2$ | $\text{cd}_1$ | S |
| 8 | $\text{sys}_2$ | $a^{(2)}$ | $\text{c}_2$ | $\text{cd}_1$ and $\text{cd}_2$ | F |

## 3.8 Conclusion

For both $\text{sys}_1$ and $\text{sys}_2$ the difficulty to learn increases along the table. The 4th and 8th are the same conditions which we desire most. Although case 4 was successful, it took much longer than cases 1,2,3 to learn. This suggests that $\text{cd}_2$ is a much harder condition to learn. Since it requires the amplitude to vanish exactly when the target fidelity is reached. I believe this is why case 8 has failed, because $\text{sys}_2$ (dim=3) is more complicated than $\text{sys}_1$ (dim=2).

I would also like to mention that although the cases that uses $a^{(2)}$ are treated as successful, the bound for $a^{(2)}$ was not small enough. Because it is observed that it becomes very hard to learn when the bound is small.