Department of Computer Science and Engineering

Undergraduate Summer Research Internship 2021

Reinforcement Learning for Autonomous Systems with Applications in Autonomous Driving

Author:
Xu Haoran
Student ID: 1155124383

Supervisors:
Dr. Han Dongkun
Mr. Huang Hejun

August 2021

Contents

1 Abstract

Reinforcement learning control has lively echoes in the control communities recently. However, due to the absence of safety guarantee, the corresponding application is still far from safety-critical scenarios, e.g., autonomous vehicles, aerospace engineering and human interactive robots. Motivated by the need of simultaneously enhancing the control performance and safety guarantee, our project aims to compare reinforcement learning controller and model predictive controller horizontally and further validate the reinforcement learning performance under the guarantee of control barrier function. Simulation results demonstrate the performance of the aforementioned method.

2 Introduction

Even with the rapid development of the learning-based control approaches, e.g., deep learning and reinforcement learning guided control, the majority industry players still prefer those classical controllers such as proportional integral derivative controller (PID) and model predictive control (MPC) controller which are completely controllable and explainable all the time. Reinforcement learning (RL) controller has good performance in simulations but the exploration and exploitation processes contain some potential dangerous actions which are definitely rejected in all the safety critical scenarios.

RL focuses on finding an agent's policy that maximizes a long-term reward, like the training for young but smart children introduced in [1] and [2], which is likely to explore unsafe actions by feeding various experiences. This method has been successfully applied in many continuous control problems, e.g., stochastic framework to approximate the dynamical system introduced in [3] and mobile robots control in [4].

Recently, combining the RL with safety guarantee over learning process, also known as the safe reinforcement learning is relaxing the application in safety-critical systems control, [5] proposed a framework to complete the safety guarantee and [6] further combined the control barrier function as part of the controller to improve the RL control policy. Previous works about the safety guarantee have utilized barrier function, [7] introduced the quadrotor controller design and [8] introduced the application in biped robot. Region of attraction can also be the safety constrain, [9] [10] introduced the approaches to regulated controller inside the region of attraction to achieve the combination of advanced approaches and safety-guarantee.

In this project, we applied the Deep Deterministic Policy Gradient (DDPG) algorithm in controller design [11] based on the RL to do comparison with the MPC controller. Then further kick off the DDPG algorithm with CBF constrain in uniting the safety guarantee and reduce the impact of the hysteresis over process.

The main contributions in this paper are:(1) compare the performance of DDPG controller and MPC controller, (2) verify the influence factors during DDPG learning process, and (3) validate the superiority of combing CBF into RL algorithm both in learning efficiency and safety guarantee via a cart-pole model.

3 Background Information

Preliminary results on Reinforcement Learning, Model Predictive Control and Control Barrier function are revisited here to set the stage for corresponding controller comparison and safe stabilization. More specifically, the aforementioned controllers inside the complex autonomous driving system and their synthesis for dynamical systems will be discussed.

3.1 Reinforcement Learning Controller

Consider a time evolution of the infinite-horizon discounted Markov decision process with control-affine, deterministic dynamics setup in discrete timestep t as

$$s_{t+1} = f(s_t) + g(s_t)a_t, \tag{1}$$

where $s_t, s_{t+1} \in \mathcal{S}$ denote the adjacent elements in the state set $\mathcal{S}$, $a_t \in \mathcal{A}$ denotes the action at state s_t inside the action set $\mathcal{A}$, f and g compose the dynamical model.

At each timestep t the system receives an observation s_t, takes an action a_t toward the next state s_{t+1} and receives a scalar reward r_t. In all the environments considered in this paper, the actions are real-valued $a_t \in I\!R^N$. Each action of system (1) is defined by a policy, π, which maps states to a probability distribution over the actions $\pi \colon \mathcal{S} \to \mathcal{P}(\mathcal{A})$.

The return from a state is defined as the sum of discounted future reward

$$J(\pi) = \mathbb{E}_{\gamma\sim\pi}[\sum_{t=0}^{T} \gamma^t r(s_i, a_i)], \tag{2}$$

with a discounting factor $\gamma \in [0, 1]$. Here $\gamma \sim \pi$ is the trajectory of system (1) as $\gamma = \{s_t, a_t, \ldots, s_{t+T}, a_{t+T}\}$ over T observations where the actions are sampled from policy π. Note that the return depends on the actions chosen on the policy π. The goal in reinforcement learning is to maximize the expected reward function (2).

The action-value function is used in many reinforcement learning algorithms. It describes the expected return after taking an action a_t in state s_t and thereafter following policy π:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i\geq t}, s_{i>t}\sim E, a_{i>t}\sim\pi}\left[r_t | s_t, a_t\right]. \tag{3}$$

Many approaches in reinforcement learning make use of the recursive relationship known as the Bellman equation:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1}\sim E}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1}\sim\pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]. \tag{4}$$

If the target policy is deterministic we can describe it as a function $\mu : \mathcal{S} \leftarrow \mathcal{A}$ for the final off-policy Q^μ:

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1}\sim E}\left[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))\right]. \tag{5}$$

RL is about how intelligent agents should react in a given state to maximize the long-term award. Differs from supervised learning and unsupervised learning, RL does not need labelled pairs data or sub-optimal actions to be explicitly corrected [12]. RL focuses on training an agent to achieve the setting goal by interacting with the state with rewards or penalties over process.

3.1.1 Deep Deterministic Policy Gradient Algorithm

Deep Deterministic Policy Gradient (DDPG) algorithm is an off-line actor-critic method that computes the policy gradient based on sampled trajectories and an estimation of the action-value function. The optimal policy can be searched by maximizing the expected cumulative long-term reward with more and more information. During the training process, the DDPG agent will update the actor and critic properties at each time step using a mini-batch of experiences randomly sampled from the experience buffer. The algorithm shown below explains how DDPG works in continuous system control.

Algorithm 1: Deep Deterministic Policy Gradient algorithm

Input: Randomly initialize weights of $Q(s, a|\theta^Q)$ and $\pi(s|\theta^\pi)$.
Output: The trained agent with optimal policy $\pi(s|\theta^\pi)$.

Initialize target net weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\pi'} \leftarrow \theta^\pi$ and replay buffer R.
for *each episode* **do**
  Initialize random process $\mathbb{N}$ for action exploration.
  Receive initial observation state s_1.
  for *each step t of episode* **do**
    Select action $a_t = \pi(s_t|\theta^\pi) + \mathbb{N}_t$.
    Execute a_t and observe reward r_t and state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in R.
    Sample from R a minibatch of $\mathbb{N}$ transitions.
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \pi'(s_i + 1|\theta^{\pi'})|\theta^{Q'})$.
    Update critic by minimizing the loss:
    $\mathbb{L} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$.
    Update the actor via sampled policy gradient:
    $\nabla_{\theta^\pi}\mathbb{L} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)\nabla_{\theta^\pi}\pi(s|\theta^\pi)|s_i$.
    Update the target networks:
    $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    $\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'}$.

[1] and [11] introduced more details about the DDPG algorithms which ensure the trained controller to operate with less hysteresis. There are several successful applications of DDPG algorithms in RL-based control system as shown in Section 2.

3.2 Model Predictive Controller

Model predictive control (MPC) is an advanced method of process control that is used in a process to satisfy a set of constraints. It is a multi-variable control algorithm that uses an internal dynamic model for all processes. Over the receding horizon, the optimization algorithm in MPC introduced in [13] is to minimize the

cost function J_m with the target control input u as follows,

$$J_m = \sum_{i=1}^{N} w_{x_i}(r_i - x_i)^2 + \sum_{i=1}^{N} w_{u_i}\Delta u_i^2. \tag{6}$$

Here x_i means i^{th} controlled variable x, r_i means i^{th} reference variable, u_i means i^{th} manipulated variable, w_{x_i} and w_{u_i} mean weighting coefficient reflecting the relative importance of x_i and weighting coefficient penalizing relative changes of u_i respectively.

Compared to the commonly used PID controller, MPC controller is more useful in handling multi-input multi-output systems that may have interactions between their inputs and outputs as [14], [15] introduced. MPC also has the ability to anticipate future events and can take control actions accordingly, more details were discussed in [16] and [17].

3.3 Barrier Function Guided Controller

Consider a control-affine dynamical system as follows,

$$\dot{x} = f(x) + g(x)u, \tag{7}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ denotes the state and $u \in \mathcal{U}$ denotes the control of the system. $f(x)$, $g(x) : \mathbb{R}^n \to \mathbb{R}^n$ are both Lipschitz continuous.

Using the invariant set principle, the control barrier function $h(x)$ (CBF) can certify that state trajectories always in the safe region $\mathcal{X}_s$ if $h(x)$ satisfies the following definition,

$$\begin{aligned} \mathcal{C} &= \{x \in \mathcal{X}_s \subset \mathbb{R}^n : h(x) \geq 0\}, \\ \partial\mathcal{C} &= \{x \in \mathcal{X}_s \subset \mathbb{R}^n : h(x) = 0\}, \\ \mathrm{Int}(\mathcal{C}) &= \{x \in \mathcal{X}_s \subset \mathbb{R}^n : h(x) > 0\}, \end{aligned} \tag{8}$$

so that the safety of the system is guaranteed, which is also summarized in the following lemma.

Lemma 1 *(Theorem 1 of [18]) For the nonlinear control system (7), if there exists a control Lyapunov function $V : D \to \mathbb{R}_{\geq 0}$, i.e., a positive definite function satisfying, then any Lipschitz continuous feedback controller $u(x)$ asymptotically stabilizes the system.*

Example 1: Consider the 2D control-affine system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + u \end{bmatrix}. \tag{9}$$

The unsafe region $\mathcal{X}_u$ is expressed by

$$\mathcal{X}_u = \begin{cases} (x_1 - 3)^2 + (x_2 - 1)^2 - 4 < 0, \\ (x_1 + 3)^2 + (x_2 + 4)^2 - 4 < 0, \\ (x_1 + 4)^2 + (x_2 - 5)^2 - 4 < 0. \end{cases} \tag{10}$$

Since system (9) is a polynomial system, the optimal barrier function $h(x)$ can be computed directly with Sum-of-squares programs using SOSOPT [19] and MOSEK [20] solver,

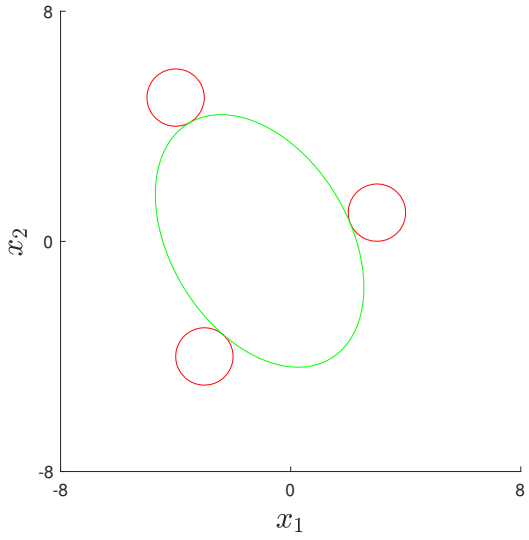


Figure 1: Estimates of safe region for a 2D system. The red circles represent the boundaries of unsafe regions. The green ellipse represents the estimate of the control barrier function certified safe region.

$$\begin{aligned} h(x) = &- 0.9346689813877013x_1^2 - 0.5685784481277874x_1x_2 \\ &- 0.6374835834694639x_2^2 - 2.009310309190127x_1 \\ &- 0.5935146025765404x_2 + 9.538241262326094, \end{aligned} \tag{11}$$

where the corresponding optimal controller is

$$\begin{aligned} u(x) = &0.05030788952646475x_1^2 + 0.4756818924140498x_1x_2 \\ &+ 0.1196003722162429x_2^2 - 1.287641305119645x_1 \\ &- 3.287641851842767x_2 - 3.09615484137644e - 06. \end{aligned} \tag{12}$$

The results for the example are shown in Fig. 1. The ellipse enclosed by green solid line denotes the optimal safe region certified by control barrier function, while the ellipse enclosed by red solid line denotes the unsafe region.

We kindly introduced interest readers to [18] for more details about the CBF which ensure the controller u regulates the state trajectory in the safe set $\mathcal{X}_s$. There are several successful applications of CBF-based learning process [6].

4 Controller Comparison between DDPG and MPC

To make comparison between DDPG agent and MPC controller in controlling continuous actions, we choose to use classic cart-pole model in Matlab. The model is built with Simscape Multibody in Matlab and shown in Fig. 2.

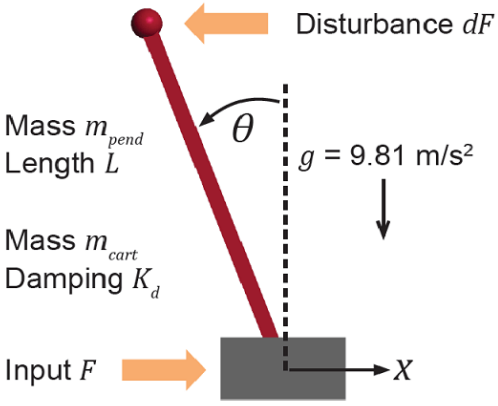


Figure 2: Cart-Pole System

This model is a pole attached to a cart moving along a frictionless horizontal track.Here we set mass of cart and mass of pendulum to be both 1 and length of pole to be 0.5.The damping is set to be 10 in our model. As we can see from Fig. 3, there are two inputs: Force F on the cart and disturbance force dF from the environment; and 4 outputs: position displacement x from the centre position of the cart, angle of displacement from the upright position of the poleθ, derivative of theta d_θ and derivative d_x for this model. At the beginning the cart-pole is stationary with $x = 0$, $\theta = 0$ and then the force F will be given to the cart as input. The control goal is to make the pole stand upright without falling over using minimal control effort, which means the outputs of the control system should be x remains unchanged and θ remains 0.

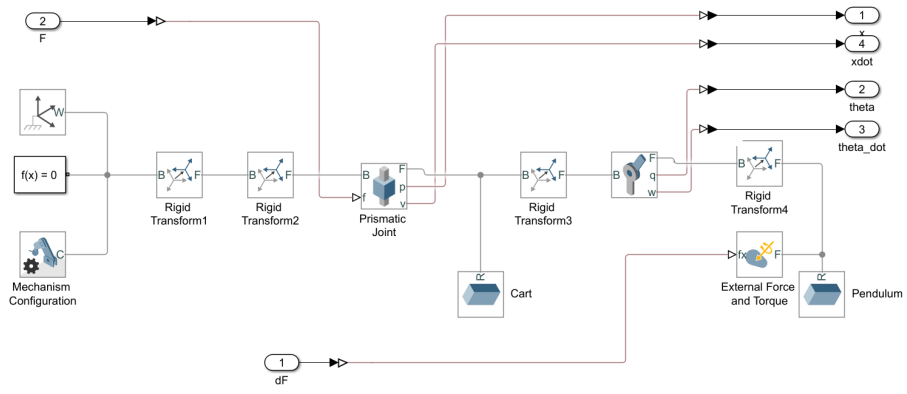


Figure 3: Cart-Pole Model Used in the Control Systems

Next we will design two control systems for DDPG and MPC controller respectively using cart-pole model.

4.1 Case Study of DDPG and MPC Control

Firstly, we design a control system to train a DDPG agent in the aforementioned cart-pole system with the help of Reinforcement Learning Toolbox and Simulink in Matlab. The structure shown in Fig. 4 shows that the system we build is a reinforcement learning system.

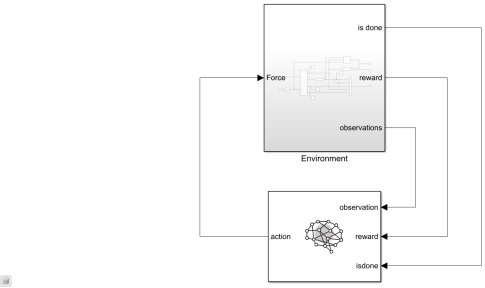


Figure 4: Structure Overview of DDPG Control System

As shown in Fig. 5, the plant here is the cart-pole model we previously built with F and dF as two inputs and x, θ, d_θ and d_x as four outputs. It's worth mentioning that F here has a boundary, and it's bounded by the upper limit and lower limit. We can change the limits here to train agents with different performance. We will go back to this topic in Session 4.3. For the reinforcement learning system, it takes $\cos\theta$, $\sin\theta$ converted from θ, together with the other three outputs from cart-pole model as the observation information. At each time step, the reward of the DDPG agent is

$$r_t = -0.1(5\theta_t^2 + x_t^2 + 0.05u_{t-1}^2) - 100B,$$

where u_{t-1} is the control effort from the previous time step and B is a flag (1 or 0) that indicates whether the cart is out of bounds. Once B hits the limit the reinforcement learning system will terminate. The action of the system is changing the force F on the cart-pole so that DDPG agent can control the system.

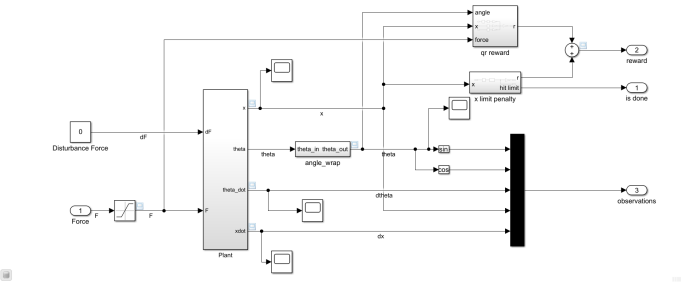


Figure 5: Detailed Structure of DDPG Control System

To compare the performance of a DDPG agent and MPC controller afterwards, we need to train our own DDPG agent first. In the previously built DDPG control system, we can start training our DDPG agent and the training results is shown in Fig. 6. Finishing training the DDPG agents, we can choose trained agent with high episode award for comparing with MPC controller afterwards.

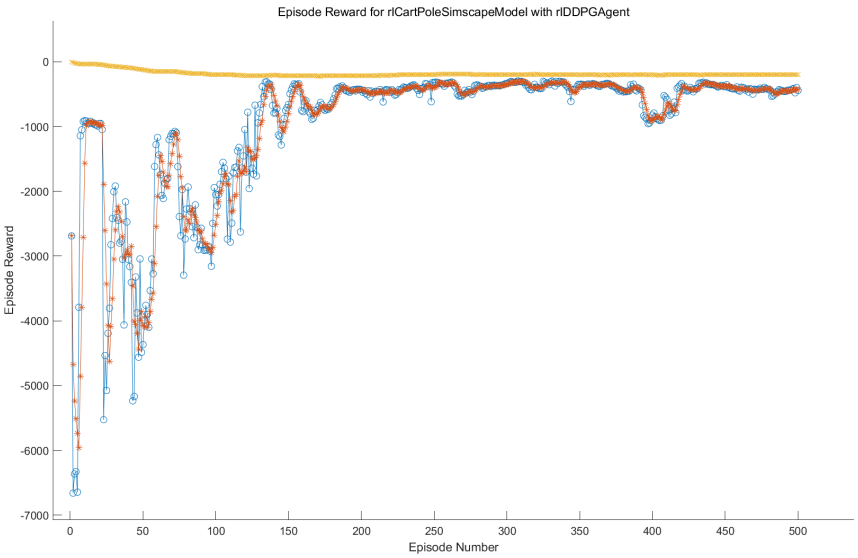


Figure 6: Training Results for DDPG Agents

Then we design a control system with MPC controller in the aforementioned cart-pole system with the help of Model Predictive Control Toolbox, Simulink Control Design and Simulink. As we can see from Fig. 7, similar to the reinforcement learning control system we designed, the cart-pole system with the same inputs and outputs is used. And we put a single MPC controller with one manipulated variable: force F, one unmeasured disturbance: disturbance force dF and two measured outputs: cart position x and pendulum angle θ in the control system. And the MPC controller will change F according to x and θ and therefore achieve the control goal.

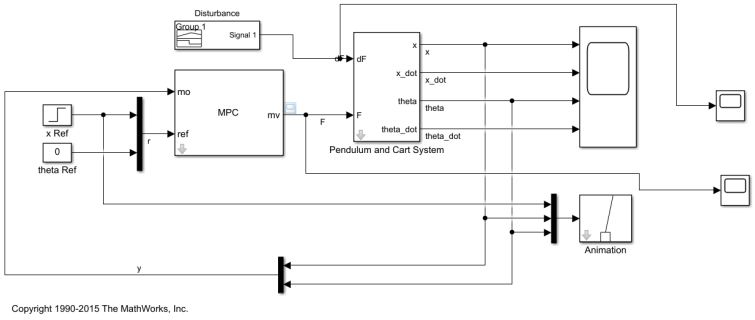
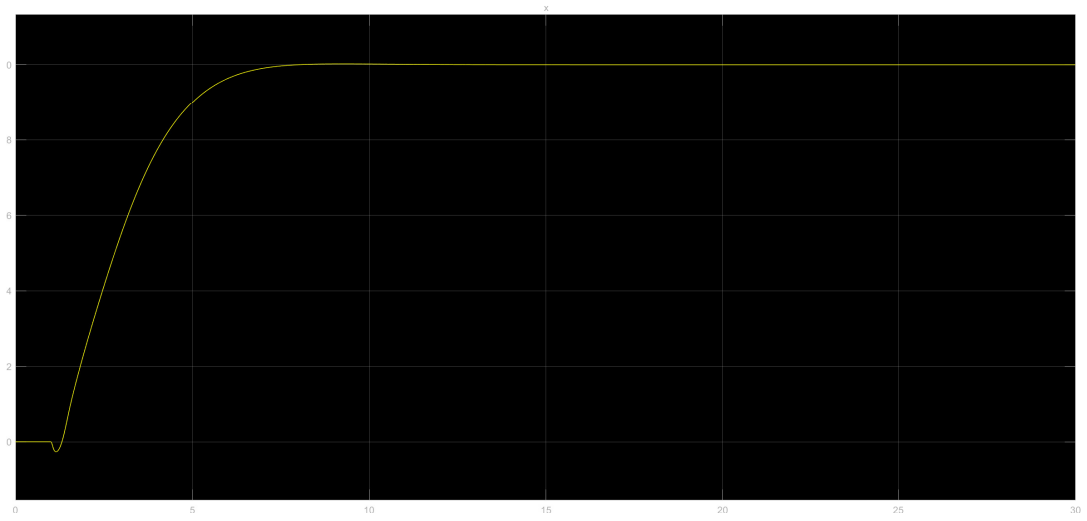


Figure 7: Detailed Structure of MPC Control System

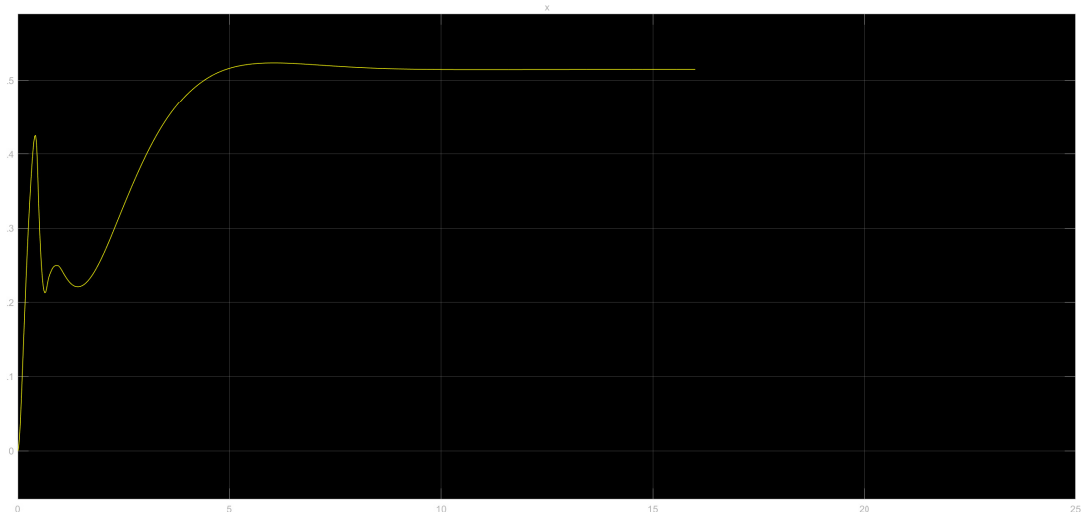
Since we use the same cart-pole model in two control systems, we can compare the outputs of x and θ from two systems to compare the time cost the achieve the control goal for the two systems. And therefore we can find out which controller has the better performance in cart-pole system control.

4.2 Outputs Analysis

After finishing training our agents, we can compare the performance of these two controllers by comparing x and θ. We first look at Fig. 8(a) and Fig.8(b), it's not hard to find out that the time cost for x to become stable(which means the cart remains stationary horizontally) is almost the same.



(a)

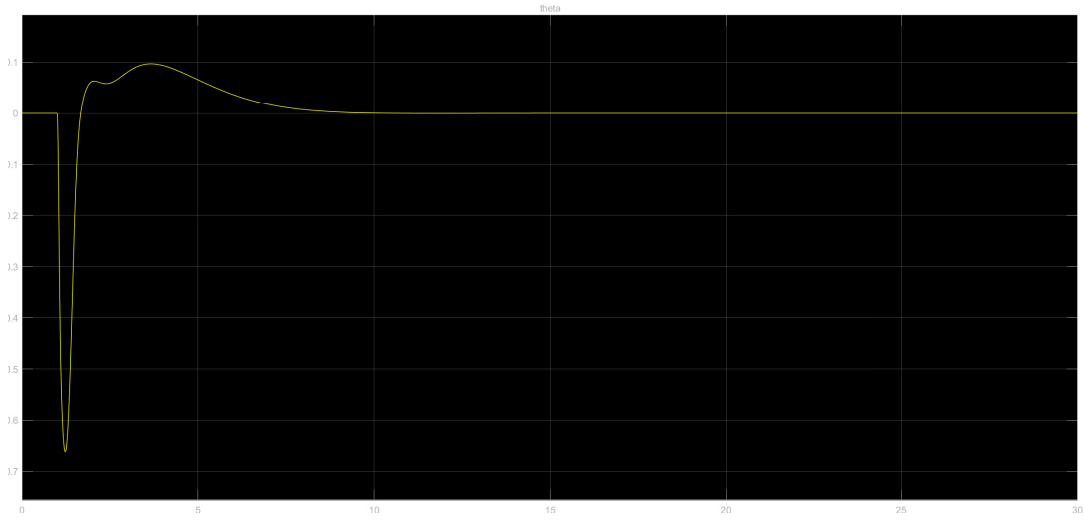


(b)

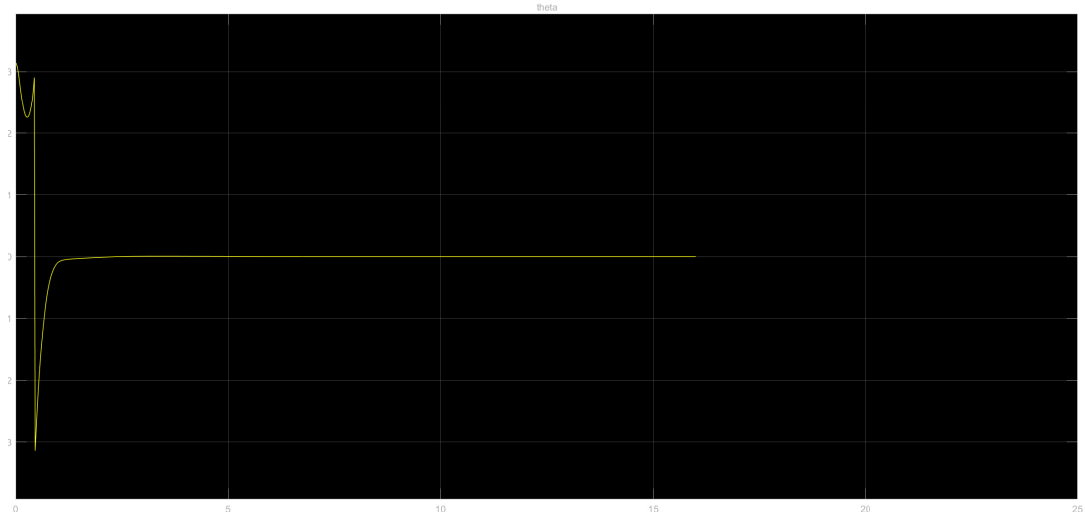
Figure 8: Output of x in MPC control System(a) and DDPG Control System(b)

However, when we look at Fig. 9(a) and Fig. 9(b), we can find out DDPG agent performs much better than MPC controller when it comes to making the pole stand upright (when θ becomes 0). The time cost of DDPG is almost one third of that by MPC controller.

Since the goal of the two control systems is to make x remains unchanged and *theta* remains 0 using minimal control effort. Therefore, we can conclude that DDPG agent has a better performance compared to MPC controller in our cart-pole system control tasks. Here we have successfully validated that DDPG agent performs better



(a)



(b)

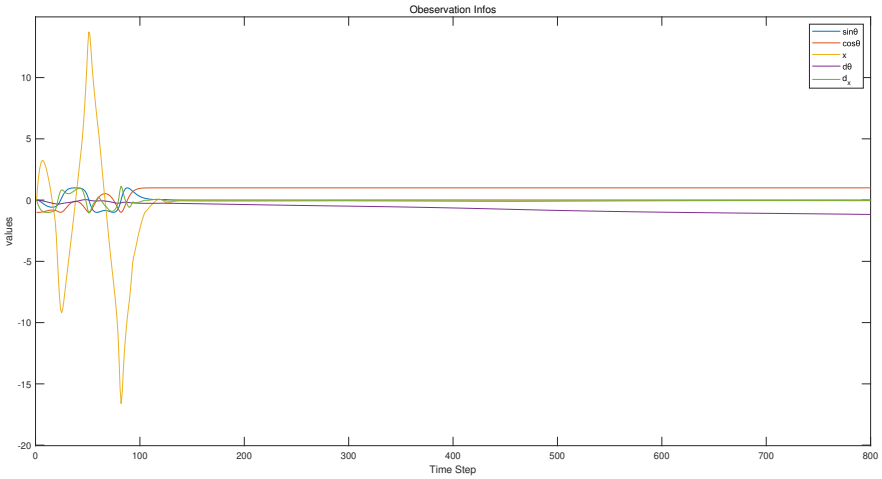
Figure 9: Output of θ in MPC Control System(a) and DDPG Control System(b)

than MPC controller in our cart-pole system.

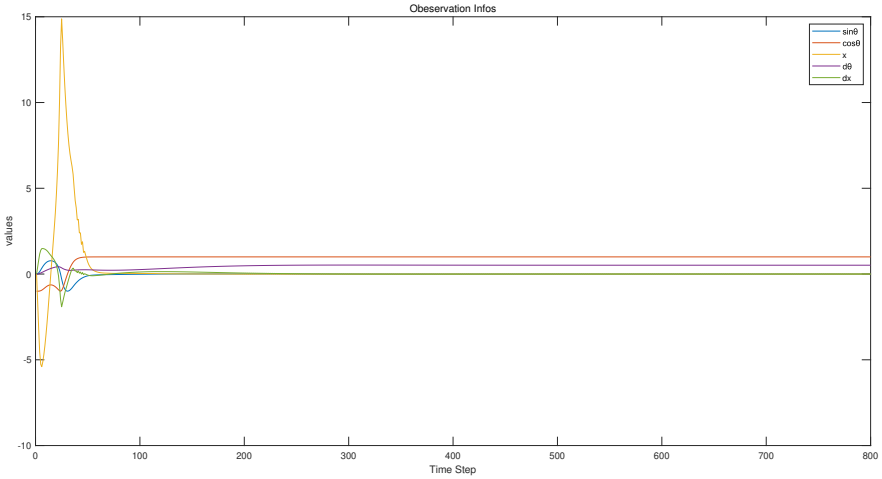
4.3 Further Analysis of DDPG

As mentioned in Session 4.1, during the process of training DDPG agent, we find out that we can further improve the performance of DDPG agent by changing the limits of input force F. We try to plot the five observation information in DDPG control system and compare them as shown in Figure 10(a) and Figure 10(b). As $\cos\theta$, $\sin\theta$ and d_θ are all determined by θ; x andd_x are all determined by x, we can observe the time cost for $\cos\theta$ to become stable at $\cos\theta = 1$ (which means $\theta = 0°$) and time cost for x to become stable to compare the performance of different agents. By observing the red line and the yellow line from two output figures, We can find out that by increasing the force limit from the default value 15 to the new value 200, the time cost to re-balance the cart-pole system can be reduced as much as a

half. Therefore, we can change the force limit accordingly in different environments to make our DDPG agent preforms better. The flexibility of DDPG agent is also an advantage over MPC controller since we can choose to use different force limit settings in different situations.



(a)



(b)

Figure 10: Observation Information with Force Limit 15(a) and Force Limit 200(b).

5 Controller Comparison between DDPG and DDPG-CBF

Though we have verified superiority of reinforcement learning agents in continuous system control compared to traditional controllers, there is still an important factor that restrict the success of reinforcement learning can reach, which is the absence of safety guarantee during the learning process. Therefore, it would be great if

we can combine the idea of learning efficiency and safety guarantee, i.e. combine reinforcement learning and CBF together.

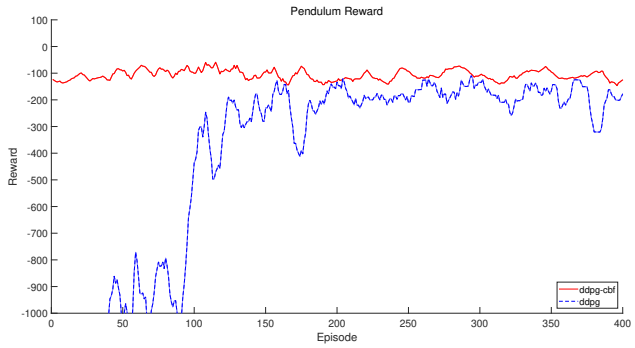
We build a control system using an inverted pendulum model from OpenAI gym environment, which has mass m and length l, and is actuated by torque u. We set the safe region to be $\theta \in [-1, 1]$ radians, and define the reward function

$$r_t = \theta_t^2 + 0.1\theta_{t-1}^2 + 0.001u_{t-1}^2,$$

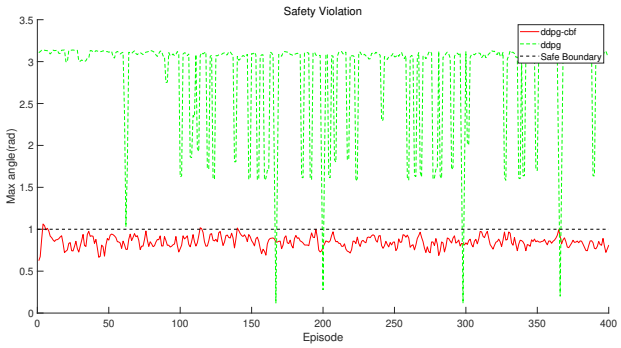
to learn a controller that keeps the pendulum upright. Here θ is the angle of pendulum from vertical and u is the control effect of the last time step. We set torque limits $u \in [-15, 15]$, $m = 1$ and $l = 1$.

5.1 Outputs Analysis between DDPG and DDPG-CBF

Firstly, we need to train the agents in DDPG and DDPG-CBF control systems. Here the training results is shown in Fig. 11(a) and clearly the episode reward of DDPG-CBF(denoted in red line) is higher than that of DDPG(denoted in blue line), which means DDPG-CBF agent can gain higher average episode reward. Besides, DDPG-CBF agent can also reach higher episode reward than DDPG agent, which means DDPG-CBF can get better long-term reward than DDPG algorithm. Therefore, the learning efficiency of DDPG-CBF is better than DDPG algorithm.



(a)



(b)

Figure 11: Comparison between DDPG and DDPG-CBF Algorithm on Episode Reward(a) and Safety Violation(b)

As the purpose of adding CBF to DDPG control system is to ensure the safety during the training process for safety-critical systems, we also plot the results of safety violation from DDPG and DDPG-CBF. Previously we set the safe region θ as [-1,1] and it's shown as the black line in Fig. 11(b). If we look at the output results from DDPG and DDPG-CBF, it's clearly to see that DDPG-CBF can keep the pendulum within the safety set we set most of the time while DDPG can barely do so. So DDPG-CBF performs better than DDPG in safety guarantee.

Therefore, the combination of DDPG and CBF has a much better performance than the original DDPG in safety violation control and also performs better in learning efficiency. In the future, we can explore the possibility of applying DDPG-CBF algorithm in more safety-critical control systems.

6 Conclusion

In this project, we have compared the performance between the DDPG controller and MPC controller in a Cart-Pole system. We also tested different system input to prove the advantages of the former one. Furthermore, we combined CBF into DDPG controller to satisfy the safety guarantee over processes. In the future, we are planning to further explore more combination of safety guarantee and RL algorithms based on the real-world equipment for advanced performance.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[3] N. Heess, G. Wayne, D. Silver, T. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," *arXiv preprint arXiv:1510.09142*, 2015.

[4] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36, IEEE, 2017.

[5] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3387–3395, 2019.

[7] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2460–2465, IEEE, 2018.

[8] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *2015 American Control Conference (ACC)*, pp. 4542–4548, IEEE, 2015.

[9] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *arXiv preprint arXiv:1705.08551*, 2017.

[10] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*, pp. 585–590, IEEE, 2018.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[13] T. A. Qin, S Joe; Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733–764, 2003.

[14] J. E. Pinsker, J. B. Lee, E. Dassau, D. E. Seborg, P. K. Bradley, R. Gondhalekar, W. C. Bevier, L. Huyett, H. C. Zisser, and F. J. Doyle, "Randomized crossover comparison of personalized mpc and pid control algorithms for the artificial pancreas," *Diabetes Care*, vol. 39, no. 7, pp. 1135–1142, 2016.

[15] M. I. Salem, Fawzan; Mosaad, "A comparison between mpc and optimal pid controllers: Case studies," in *Michael Faraday IET International Summit 2015*, IET, 2015.

[16] C. Kreuzen, "Cooperative adaptive cruise control: using information from multiple predecessors in combination with mpc," 2012.

[17] J. B. Jørgensen, J. K. Huusom, and J. B. Rawlings, "Finite horizon mpc for systems in innovation form," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 1896–1903, IEEE, 2011.

[18] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[19] P. Seiler, "Sosopt: A toolbox for polynomial optimization," *arXiv preprint arXiv:1308.1889*, 2013.

[20] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, version*, vol. 4, 2019.