Soft Actor Critic Controller Integrated with Barrier State

Interim Report for Progress Meeting

MinGyu Kim

1 Introduction

Currently, many control systems rely on the dynamics model. As the system develops to be more complex for acquiring more diverse actions, it has become more important to know the exact model and prioritize its safety. There are many methods that help the control system to know the model, but this paper focuses on Reinforcement Learning (RL) that finds optimal control without the exact model dynamics. Nevertheless, there are some disadvantages of using reinforcement learning, one of which is that the agent can violate environmental constraints while being trained. In a real environment, there are many obstacles around the agent, some of which are moving uncertainly and some of which are fixed at specific locations. In this paper, the way an agent trained by the RL avoids static obstacles is discussed, and it presents how to train an agent in a safety-critical way alongside using the RL controller.

2 Method

2.1 Soft Actor Critic Controller

As an RL controller, Soft Actor Critic algorithm was selected as it finds the control better through stochastic policy optimization. The SAC algorithm has a critical feature, entropy regularization, which ”maximizes a trade-off between expected return and entropy, a measure of randomness in the policy. This has a close connection to the exploration-exploitation trade-off: increasing entropy results in more exploration, which can accelerate learning later on. It can also prevent the policy from prematurely converging to a bad local optimum” [1]. It was thought that the entropy variable can adjust the extent of how more often the agent should experience the penalty cost due to the barrier function and barrier state. Therefore, it was expected that the entropy variable would play a key role in finding a point where an agent experiences a proper amount of penalty.
The entropy function is computed with a probability density function P as

$$H(P) = \underset{x \sim P}{E}[-logP(x)]. \tag{1}$$

The optimal policy is computed with the regularized entropy variable to receive a bonus reward at time step like Equation 2.

$$\pi^* = \underset{\pi}{argmax} \underset{\tau \sim \pi}{E}[\Sigma_{t=0}\gamma^t(R(s_t, a_t, s_{t+1} + \alpha H(\pi(-|s_t))))] \tag{2}$$

where α > 0 is the trade-off coefficient and R is reward.
Therefore, the state value function can be reduced to

$$V^\pi(s) = \underset{a \sim \pi}{E}[Q^\pi(s,a)] + \alpha H(\pi(-|s)), \tag{3}$$

and the action-value function can be reduced to

$$Q^\pi(s,a) = \underset{s' \sim \pi}{E}[R(s,a,s') + \gamma(Q^\pi(s',a') + \alpha H(\pi(-|s')))] = \underset{s' \sim P}{E}[R(s,a,s') + \gamma V^\pi(s')]. \tag{4}$$

2.2 Barrier State (BaS)

For building a safety-critical system, another extra state variable is introduced, which is called to represent as h(x) = constraint equation. This new variable is evaluated as a measure of safety distance between the agent and obstacle. The barrier state operator **B**, which exploits the constraint equations, is expressed as either of the following equations [2],

$$\mathbf{B} = \frac{1}{h(x)} \tag{5a}$$

$$\mathbf{B} = -log\frac{h(x)}{1+h(x)}. \tag{5b}$$

The dynamics of the operator is written as

$$\dot{B} = \frac{\delta B}{\delta h}\frac{\delta h}{\delta x}\frac{\delta x}{\delta t} = \dot{B}h_x\dot{x} = \dot{B}h_x(f(x) + g(x)u) \tag{6}$$

where $\dot{x}$ = model dynamics = f(x) + g(x)u (control affine system).
The safety constraint equation derived from the barrier state operator is written as by zeroing it

$$\dot{z} = \dot{\beta}h_x\dot{x} - \gamma * (z - (\beta - \beta_o)). \tag{7}$$

where **B** from Equation 6 is replaced by β.

When multiple constraints are considered, all constraint equations are summed up into one equation, like the following equation,

$$\mathbf{B} = \beta = \Sigma_{i=1}^{k}\frac{1}{h_i} \tag{8a}$$

$$\mathbf{B} = \beta = \Sigma_{i=1}^{k} - log\frac{h(x)}{1+h(x)}. \tag{8b}$$

Depending on the operator chosen in Equation 5, a different summation is required.

3 Results

3.1 SAC-BaS Optimal Control States

All simulations were implemented in the dubins vehicle environment with the maximum episode of 200, as the length of the episode was enough to see a significance of the performance difference with different controllers. The states of the vehicle at the final episode are plotted in Figure 1.

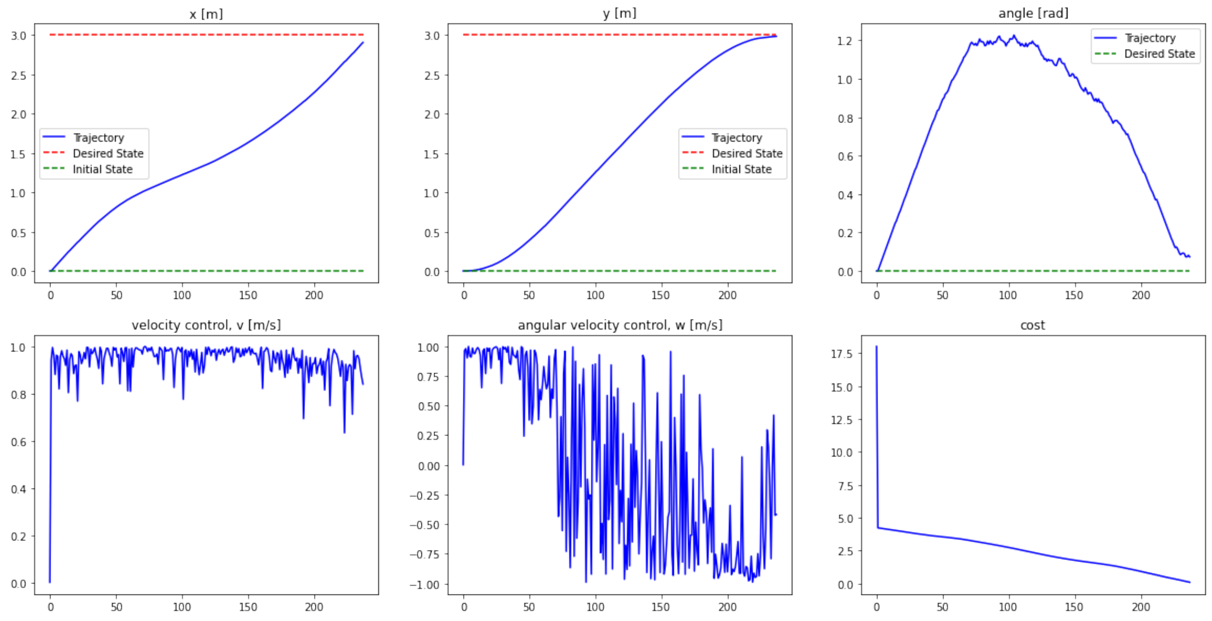


Figure 1: The state trajectory generated by the optimal control at the final episode.

The corresponding Barrier State value is plotted in Figure 2.

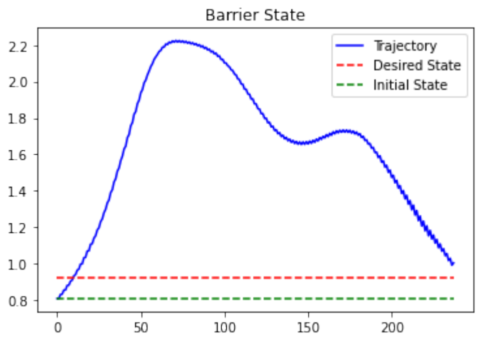


Figure 2: The Barrier State value corresponding to the state variables at the final episode.

All necessary states successfully converged to the desired state as the cost decreased to about zero.

3.2 Trajectory

The visualization of the environment and final trajectory is presented in Figure 3. The agent stably, smoothly reached to the destination without any collision (no violation).

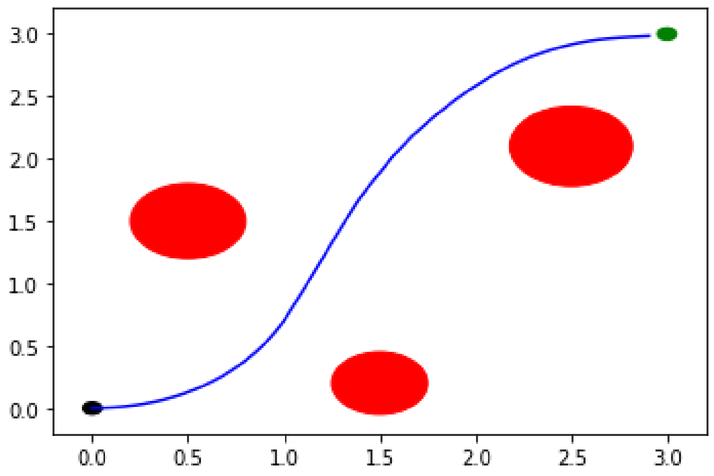
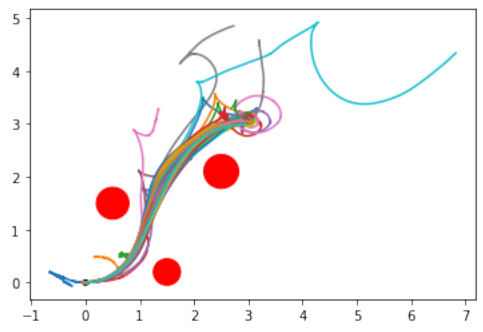
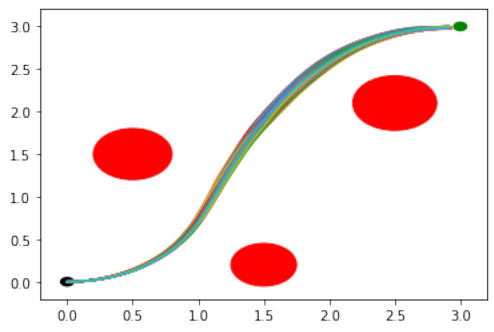


Figure 3: The trajectory generated by the SAC-BaS at the final episode. The black circle is the initial state, the green circle is the desired state, and the red circles are obstacles.

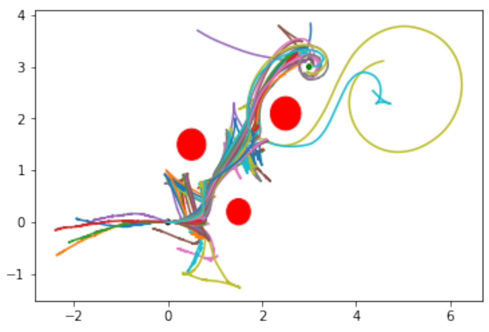
Another extra graphical work was done to show the progress made at each episode by both the SAC-BaS and SAC-CBF (Control Barrier Function). As seen in Figure 4, both controllers took many different actions at initial episodes. However, the SAC-CBF controller showed a little rough curves and tended to lose the path when it faced the obstacle in front of itself. On the other hand, the SAC-BaS controller that was updated at every step with feedback from the cost due to the BaS value consistently presented smoother and clearer trajectories.



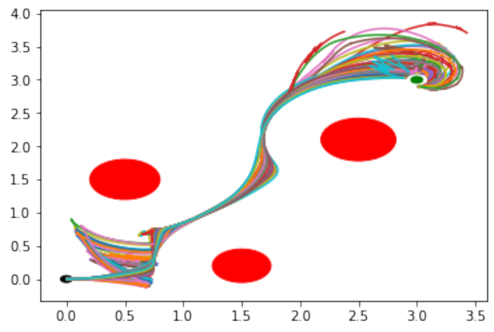
((a)) SAC-BaS at episodes [1, 100]



((b)) SAC-BaS at episodes [101, 200]



((c)) SAC-CBF at episodes [1, 100]



((d)) SAC-CBF at episodes [101, 200]

Figure 4: The visualizations of the trajectories computed by the SAC-BaS and SAC-CBF controllers.

3.3 Reward Comparison

As implied in the trajectory visualization, the SAC-BaS performed much better than the SAC-CBF in the same environment within the same length of the episode.

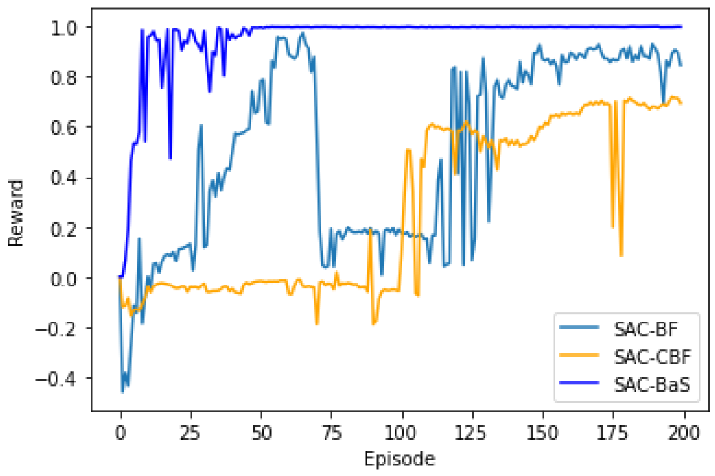


Figure 5: Performance comparison: Normalized rewards with the SAC-BF, SAC-BaS, and SAC-CBF controllers.

4 Short Discussion

1. Where the performance improvement/difference come from - mathematical analysis needed. Currently, this is being discussed. The SAC-CBF controller collects the safe action that generates the path without collision, while the SAC-BaS controller receives a direct feedback from the BaS value integrated with the cost value. I am now thinking and discussing how to interpret this and how to explain it in a mathematical way.

2. OpenAI Gym/Brax and SAC-BaS
   For publication of this paper, Hassan and I are planning on implementing the controller in the different virtual environments throughout OpenAI Gym or Brax.

3. SAC-BaS application for different environment and with moving obstacles.
   We will send and run the trained agent in different environments and see how it behaves against the obstacles. Also, we are thinking to see their performances in an uncertain environment where obstacles are moving.

4. There will be more discussions to be considered as we plan on transplanting the controller to the virtual environment now.

References

[1] Josh Achiam. *Soft Actor-Critic*. https://spinningup.openai.com/en/latest/algorithms/sac.html, 2018.

[2] Hassan Almubarak, Nader Sadegh, and Evangelos A. Theodorou. Safety embedded control of nonlinear systems via barrier states. *IEEE Control Systems Letters*, 6:1328–1333, 2022.