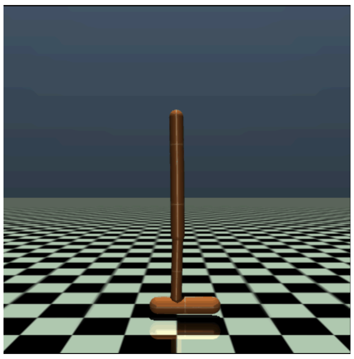
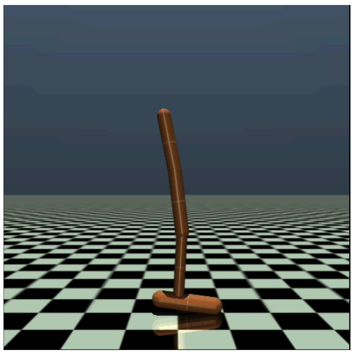
HW3: Model-based RL in “Hopper” environment

Description

In this task, we aim to increase the number of independent state and control variables compared to classical control environments. The hopper is a two-dimensional one-legged figure consisting of four main body parts - the torso at the top, the thigh in the middle, the leg at the bottom, and a single foot on which the entire body rests. The goal is to make hops that move in the forward (right) direction by applying torque to the three hinges that connect the four body parts.





Action Space

- 3-dimensional continuous space, range $[-1, 1]$
- An action represents the torques applied at the hinge joints.

Observation Space

- 11-dimensional continuous space, range $[-\infty, \infty]$
- Includes position values and velocities of various body parts

Rewards

The total reward is: **reward** = healthy_reward + forward_reward - ctrl_cost.

- healthy_reward: Every timestep that the Hopper is healthy, it gets a reward of fixed value `healthy_reward`.
- forward_reward: Reward for moving forward
- ctrl_cost: Penalty for taking large actions

Note

For detailed information, refer to: Hopper - Gymnasium Documentation

Task 1 [45%]

Implement “Model-based RL v1.5” algorithm in “Hopper” environment.

Task 2 [45%]

Implement **MBPO** (Model-Based Policy Optimization, arXiv:1906.08253) algorithm in "Hopper" environment.

Task 3 [10%]

Having trained with both algorithms, provide a comprehensive comparison between the “Model-based RL v1.5” and “MBPO” implementations. Compare the following aspects:

1. **Learning Efficiency**: Evaluate the efficiency of learning convergence for each algorithm, including the number of episodes or training steps required to achieve a satisfactory level of performance.
2. **Sample Efficiency**: Assess the sample efficiency of each algorithm by comparing the amount of interaction with the environment needed to achieve a certain level of performance.

3. **Stability**: Analyze the stability of learning for each algorithm, considering factors such as training variance and robustness to hyperparameter settings.
4. **Performance**: Compare the final performance achieved by each algorithm in terms of task-specific metrics (e.g., average return, final policy performance).
5. **Computational Resources**: Discuss the computational resources required for training and inference, including CPU and GPU utilization, memory usage, and training time.

Submission

You can use either gym [old version] or gymnasium [new version].

Draw a reward plot.

Screen record or Video record the results by rendering for at least **60 seconds**.

Pack your report with all requirements above in zip file and submit to Balckboard before April 23, 23:59pm.

```
Homework1_ID_Name.zip → task1 → Code.py [code file]
                         |        Plot.jpg [image file]
                         |        Video.mp4 [video file]
                         → task2 → Code.py [code file]
                         |        Plot.jpg [image file]
                         |        Video.mp4 [video file]
                         → report.pdf [report file]
```