CMU 10-703 Homework 3 Report

Zhengkun Dai, Xiaoyu Bai
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213
zhengkud@andrew.cmu.edu, *xiaoyub@andrew.cmu.edu*

Problem 1

LQR

The theory of optimal control is concerned with operating a dynamic system at minimum cost. Specifically, a linear-quadratic-regulator (LQR) means dynamic is linear and cost is quadratic.

LQR algorithm can be divided into 4 types: finite-horizon and continuous-time, infinite-horizon and continuous-time, finite-horizon and discrete-time, infinite-horizon and discrete-time. According to the function template signature, we choose to use infinite-horizon and continuous-time [1].

In the case of infinite-horizon and continuous-time, the dynamics can be described by:

$$\frac{dx}{dt} = Ax + Bu$$

where x is the position and u is the control. The cost can be described as:

$$J = \int ((x_{current} - x_{goal})^T \cdot Q \cdot (x_{current} - x_{goal}) + u^T \cdot R \cdot u)\, dt$$

The optimal feedback control from current position to goal position is given by:

$$u_{optimal} = -K \cdot (x_{current} - x_{goal})$$

where K is given by:

$$K = R^{-1} \cdot B^T \cdot P(t)$$

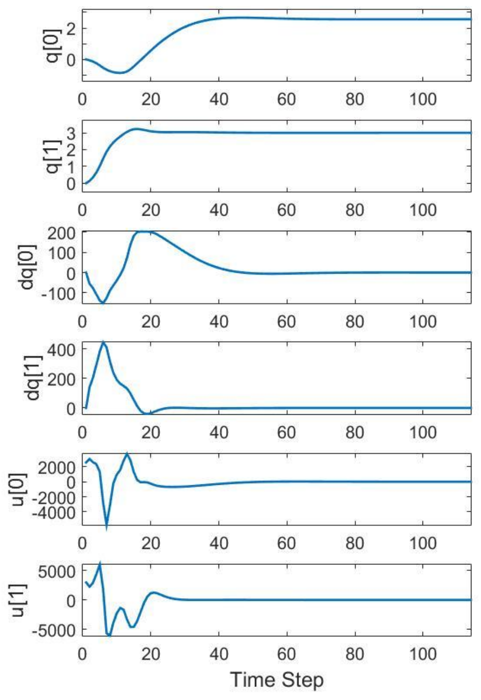
and $P(t)$ is found by solving the continuous time algebraic Riccati equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

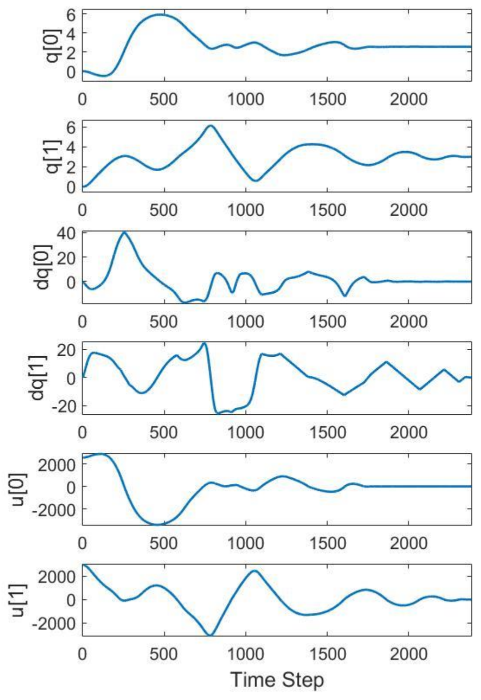
In practice, we use Newton difference method to approximate system linear parameter A and B at each time step. Although the estimation for A and B is not supposed to change much for different states. If A or B vary a lot for different time steps, it means either the system is highly non-linear, or the approximation method is wrong.

The 4 implementations for this section is quite similar. The only difference is the environment. The programming results are shown as following.

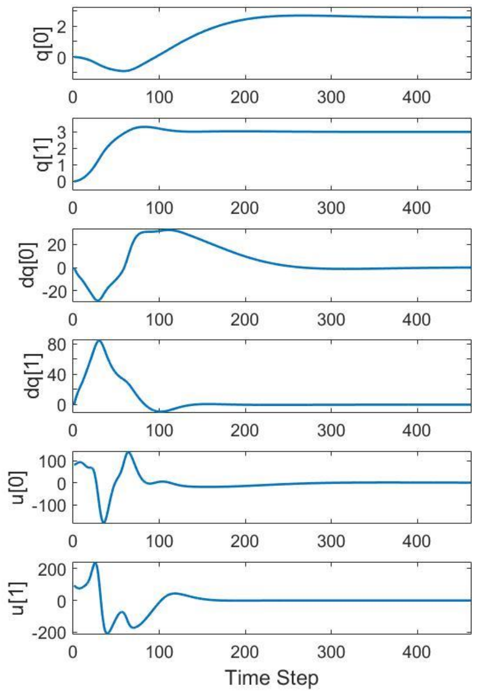
"TwoLinkArm-v0"



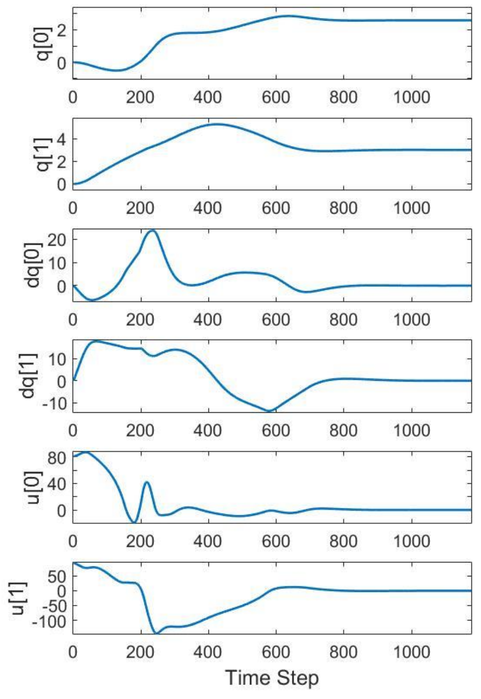
"TwoLinkArm-limited-torque-v0"



"TwoLinkArm-v1"

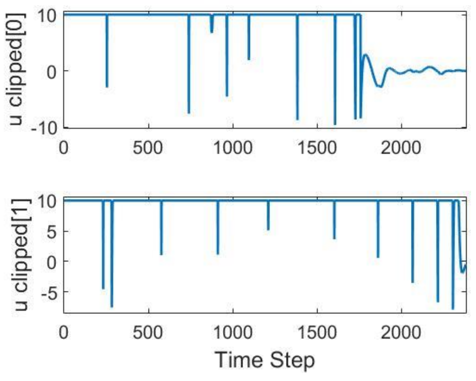


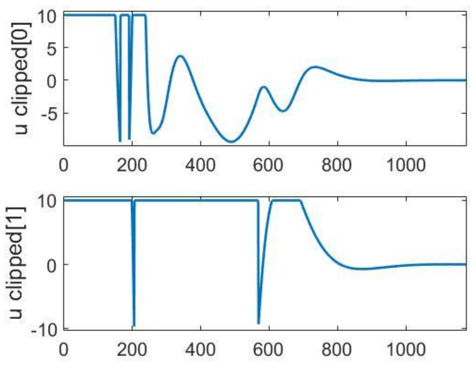
"TwoLinkArm-limited-torque-v1"



Clipped control plot.

"TwoLinkArm-limited-torque-v0" ***"TwoLinkArm-limited-torque-v1"***





Q3. Comparison between TwoLinkArm and TowLinkArm-limited-torque.

Since we are using infinite horizon LQR, the total time step number is not previously determined. Normal case converges much faster than limited-torque case. In limited-torque case, the torque is clipped and it is not the actual optimal value to control system. Therefore we see some round-about movement from robotic arm before reaching the goal position. These round-about movement lead the system to converge much more slowly than normal. From the plot of clipped control, we can see that at the beginning, most of the control torques have been clipped. So this should be the reason of slow convergence.

Q6. Comparison between normal case and high-cost-on-control case.

By looking at the *u*-plot which is the magnitude of control torque, we see that high-cost-on-control case has very small control torque compared with normal case. This is also why *"TwoLinkArm-v1"* converges more slowly than *"TwoLinkArm-v0"*. On the contrary, for the clipped cases, high-cost-on-control will reduces some incorrect big-step movement therefore the convergence is faster. Basically these two applies to different situations: one for time-efficient, and one for energy-efficient.

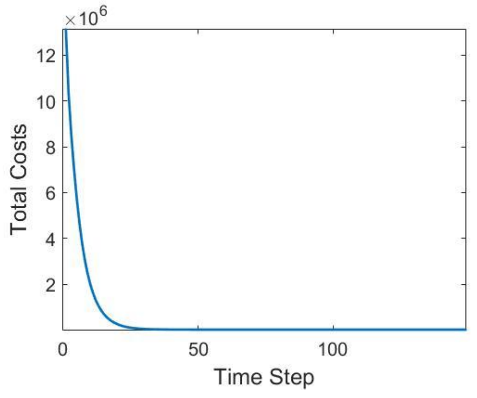
iLQR

In the algorithm of LQR, we are optimizing the current step only without caring about the future consequences. By comparison, the algorithm of iterative LQR (iLQR) plans out a trajectory that optimizes the overall cost for the entire process. The algorithm of iLQR is especially better than LQR when applied to non-linear systems.

As for the cost function, we use *inter_scale=1e-2* and *final_scale=1e6*. Our understanding is that: if final cost weighs too much, the robotic arm will reach the goal state but the trajectory may not be optimized; if the intermediate cost weighs too much, the robotic arm will go along an optimized trajectory yet the final state may not be accurate as the goal state. Therefore it is critical to balance the ratio between final cost and intermediate cost in the cost function.

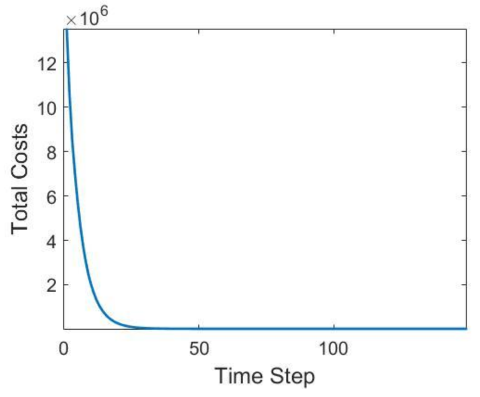
"TwoLinkArm-v0"

Cost function:

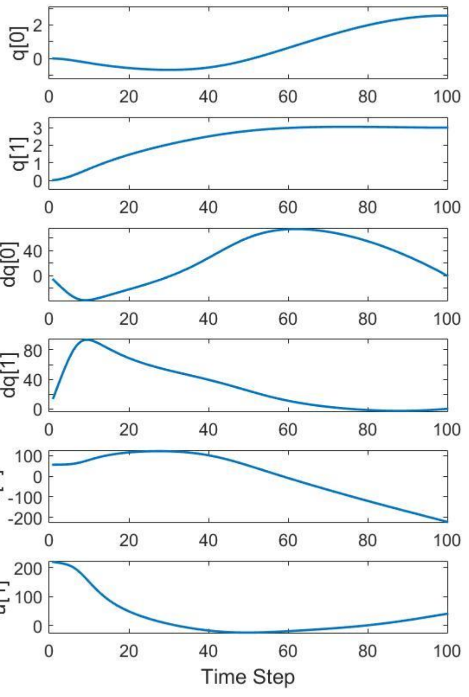


"TwoLinkArm-v1"

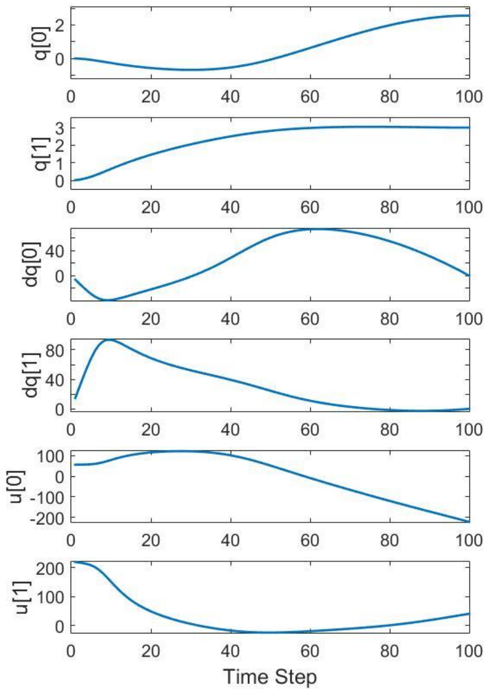
Cost function:



"TwoLinkArm-v0"



"TwoLinkArm-v1"



There is almost no difference between environment *v0* and *v1*, meaning iLQR algorithm has already optimized control amplitude without need to explicitly put high penalty on large control torque.

Q3. Comparison between LQR and iLQR

In terms of final results, iLQR is much better than LQR since it optimized over the entire moving process, while LQR only focuses on the current step. The algorithm of iLQR converges faster than LQR, showing a better trajectory than LQR without need to explicitly put high cost on control input. The only disadvantage for iLQR is higher computational expense.

Extra Credit: Speed up iLQR

We tried changing cost function to speed up iLQR.

Default: *inter_scale=1* and *final_scale=1e4*.
The goal state is *q: [2.56, 3], dq: [0, 0]*.
Our final state is *q: [-0.41026612, 0.9600892], dq: [0.1289747, 0.09374372]*.
It takes 118 steps to converge.
So we found out that instead of going to the goal state, it stops at an intermediate position and stop moving. This is due to too much weight is set to the intermediate cost. The agent would rather save the movement instead of reaching the goal state. So we should put more weight on final cost and less weight on intermediate cost.

Less weight for intermediate cost: *inter_scale=1e-2* and *final_scale=1e4*.
The goal state is *q: [2.56, 3], dq: [0, 0]*.
Our final state is *q: [2.25, 3.08], dq: [1.33e-02, -6.04e-04]*.
It takes 139 steps to converge.
This is better compared with the default value although it takes more steps to converge.

Less weight for intermediate cost and more weight on final cost:
inter_scale=1e-2 and *final_scale=1e6*.
The goal state is *q: [2.56, 3], dq: [0, 0]*.
Our final state is *q: [2.56, 3.00], dq: [1.44e-04, -4.21e-05]*.
It takes 100 steps to converge. So it seems if we put more weight on final cost and less weight on intermediate cost, we get both better performance and faster convergence.

Problem 2

1. Final loss and accuracy in training

We train the learner for 200 epochs. And use the default learning rate in ADAM optimizer.

	Final Loss	Final Accuracy
Expert Epoch 1	0.423973	0.84375
Expert Epoch 10	0.441526	0.8125
Expert Epoch 50	0.271361	0.90625
Expert Epoch 100	0.390403	0.8125

2. Evaluate cloned model

We evaluate each cloned model for 100 times and get the average reward.

	Learner Average Reward
Expert Epoch 1	173.32
Expert Epoch 10	200
Expert Epoch 50	200
Expert Epoch 100	200

For this easy environment, the learner can almost do as good as the expert except for the case in which expert only proceed 1 epoch to train the learner.

3. Evaluate cloned model in hard environment

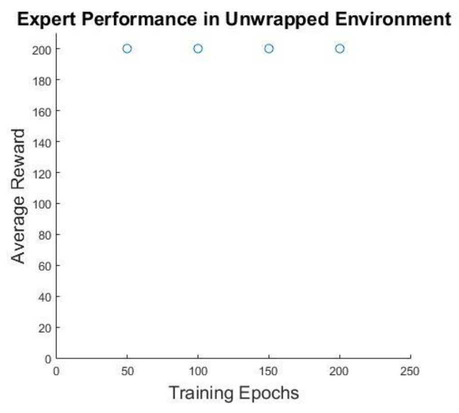
	Expert Average Reward	Learner Average Reward
Expert Epoch 1	64.99	14.76
Expert Epoch 10	63.11	74.98
Expert Epoch 50	65.99	46.1
Expert Epoch 100	65.16	106.31

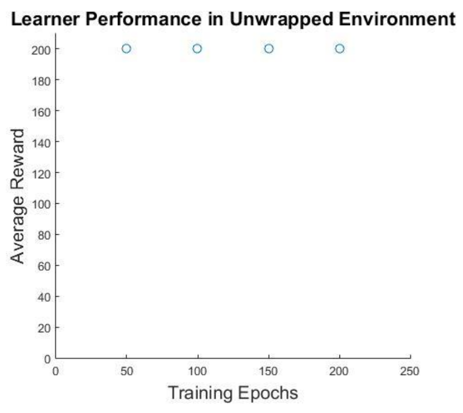
The trend is that the more epoch expert model has proceeded to generate training data, the better learner model will perform in the hard environment. When training epoch is enough, the learner can actually do better than the expert.

Extra Credit: DAGGER

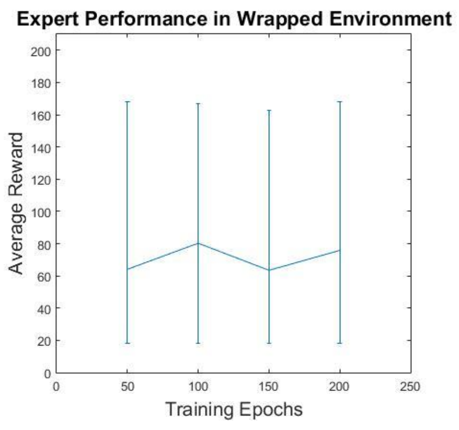
1. Learner's Performance in Unwrapped Environment

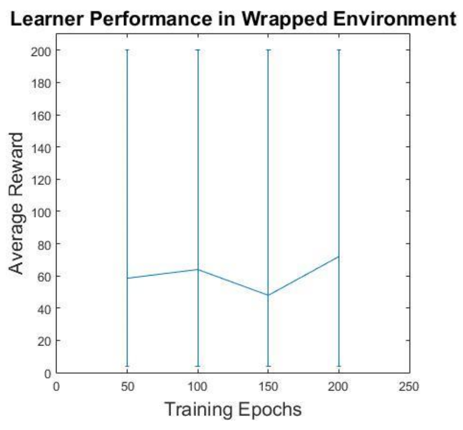
We trained the learner for 50, 100, 150, 200 epochs using DAGGER. For the unwrapped environment, the learner model can achieve 200 reward for all the 100 repetitions. So there is no meaning to plot error bar for this data.





2. Learner's Performance in Wrapped Environment



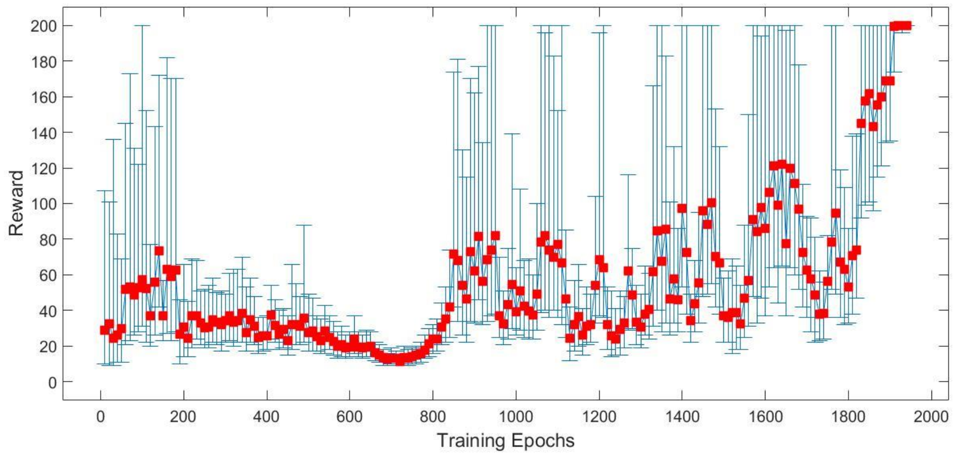


In hard environment, learner's performance is similar to expert performance but with larger standard deviation. Learner's performance in behavior cloning is generally worse than DAGGER except for only one case of 200 training epochs, the best performance is 106.31 which is better than DAGGER which is 71.96.

Problem 3

1. Learning Curve of Agent

We evaluate our model for every 10 episodes and run 100 times of the current policy to evaluate the model.



2. Comparison with previous expert and cloned model

After about 1900 training epochs, this policy converges and stabilize at reward=200. However in behavior cloning, the learner model can reach this level only after 10 epochs. This is due to the guidance of the expert. With supervised learning, agent is able to learn much faster than learning from scratch.

References

[1] https://en.wikipedia.org/wiki/Linear-quadratic_regulator.