Optimal and Learning Control for Robotics
Exercise Series 2

Bo Peng, New York University

Spring 2020

Exercise 1

Solution a) An uncontrolled system is stable if and only if all of the eigen-values of its dynamic matrix A satisfy $|\lambda_i| < 1$.

The eigen-values of the 3 systems are shown below shown below:

Table 1: Eigen-Values of the System

	λ_1	λ_2	λ_3
System 1	1.0	0.25+1.39194 i	0.25-1.39194 i
System 2	0.87744	0.06128+0.37243 i	0.06128-0.37243 i
System 3	2.0	1.41421 i	-1.41421 i

where i is the imaginary unit.

For the system with complex eigen-values, it's stable if and only if all of its eigen-values have norm strictly less than one. So, for the system 1 we don't know whether it's stable because it has an eigen-value $\lambda_1 = 1$; for the system 2, it's stable because all of its eigen-values have norm less than 1; and the system 3 is not stable for all its eigen-values have norm larger than 1.

b) A system is controllable if and only if the matrix

$$\begin{bmatrix} B & AB & A^2B & ... & A^{k-1}B \end{bmatrix} \tag{1}$$

has full row rank, where k is the dimension of vector x_n.

For the system 1, the matrix is:

$$\begin{bmatrix} 0. & 0. & 0. & 0.5 & 1. & 0.75 \\ 1. & 0. & 0. & -2. & -4. & -2. \\ 0. & 1. & 2. & 1. & 2. & -1. \end{bmatrix} \tag{2}$$

The rank of this matrix is 3, so system 1 is controllable.

For the system 2, the matrix is:

$$\begin{bmatrix} 0. & 0. & 0. & 0.5 & 0.25 & 0.5 \\ 1. & 0. & 0. & -0.5 & -0.25 & -0.25 \\ 0. & 1. & 0.5 & 0.5 & 0.25 & 0.25 \end{bmatrix} \tag{3}$$

The rank of this matrix is 3, so system 2 is controllable.
For the system 3, the matrix is:

$$\begin{bmatrix} 0. & 0. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & -2. & -2. & 0. \\ 0. & 1. & 1. & 0. & 0. & -2. \end{bmatrix} \tag{4}$$

The rank of this matrix is 2, so system 3 is not controllable.

c) If the system is controllable, LQR could make the system stable for infinity horizons. So LQR could drive the system to the origin from any initial conditions for the system 1 and system 2 for infinity horizons, but there's no such guarantee for the system 3.

d) The python code to compute the optimal control policy is shown here:

```python
def LQR(A, B, Q, R, N):
    """
    Solve the LQR with given system and cost function.

    Args:
        A: the system dynamics;
        B: the control matrix;
        Q: the quadratic cost matrix, x.T Q x;
        R: the quadratic cost matrix, u.T R u;
        N: the number of time step.

    Return:
        Ks: the gains.
    """

    def DARE(P):
        """
        A helper function to solve discrete-time algebraic Riccati equation (DARE)
        for each time step.
        """

        K = -np.linalg.inv(B.T @ P @ B + R) @ B.T @ P @ A
        P = Q + A.T @ P @ A + A.T @ P @ B @ K

        return K, P

    Ks = []

    P = Q
    for i in range(N):
        K, P = DARE(P)
        Ks = [K] + Ks

    return Ks

```

Listing 1: LQR Algorithm

In this case, the parameters are:

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{5}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6}$$

$$N = 20 \tag{7}$$

For the system 1, the optimal gain is:

$$K_1 = \begin{bmatrix} -0.03676 & -0.00031 & 1.94464 \\ -4.11166 & -1.98583 & -1.13261 \end{bmatrix} \tag{8}$$

And the simulation results for the uncontrolled and the controlled system are shown below:

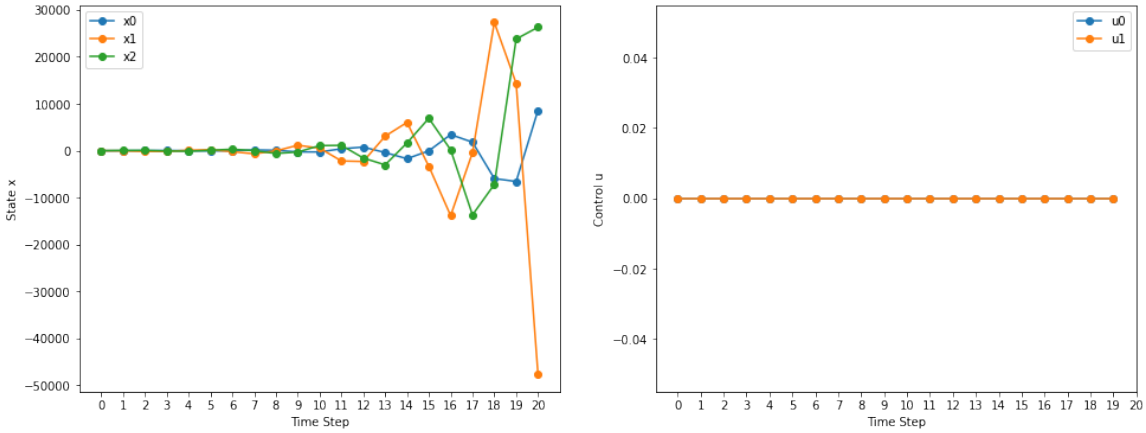

Figure 1: Uncontrolled System 1

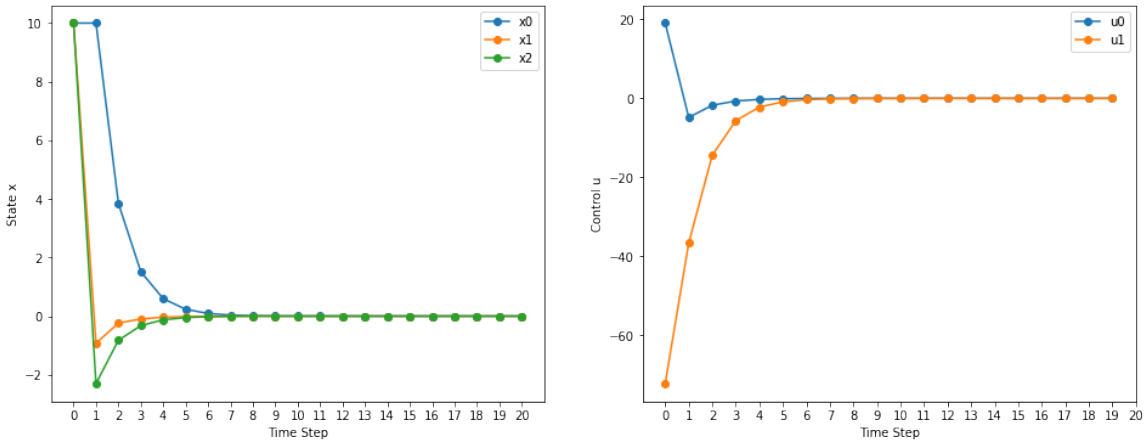

Figure 2: Controlled System 1

For the system 2, the optimal gain is:

$$K_2 = \begin{bmatrix} -0.00117 & -0.00001 & 0.49389 \\ -0.61444 & -0.49623 & -0.61443 \end{bmatrix} \tag{9}$$

And the simulation results for the uncontrolled and the controlled system are shown below:

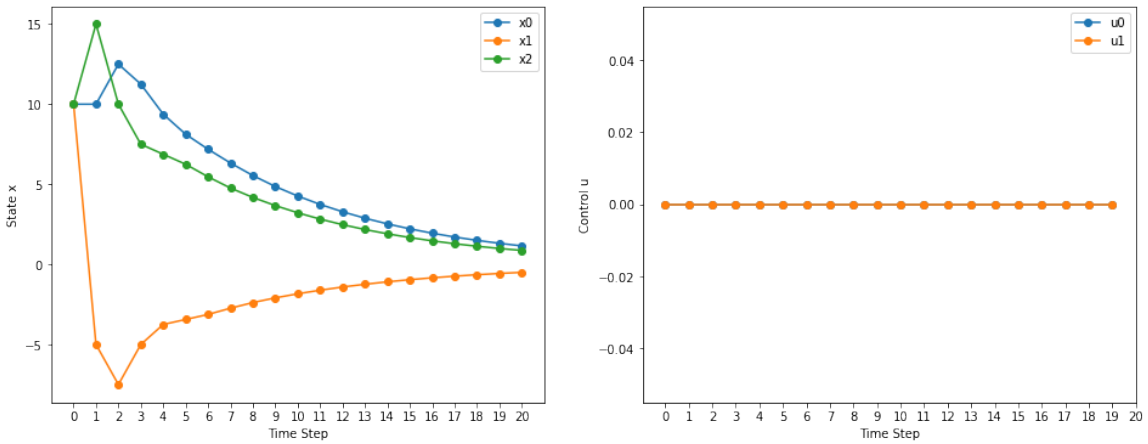

Figure 3: Uncontrolled System 2

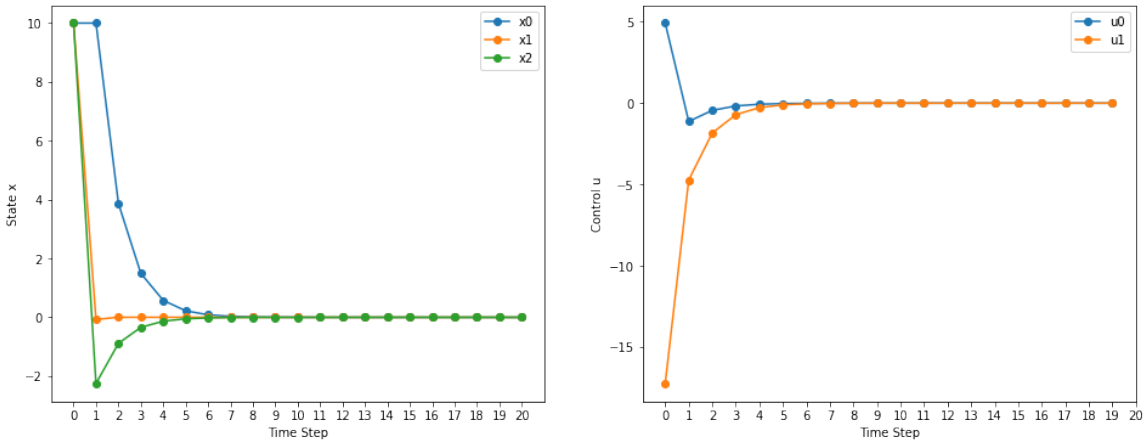

Figure 4: Controlled System 2

For the system 3, the optimal gain is:

$$K_3 = \begin{bmatrix} -0.01941 & 0. & 1.98039 \\ -0.98973 & -0.99047 & 0. \end{bmatrix} \tag{10}$$

And the simulation results for the uncontrolled and the controlled system are shown below:

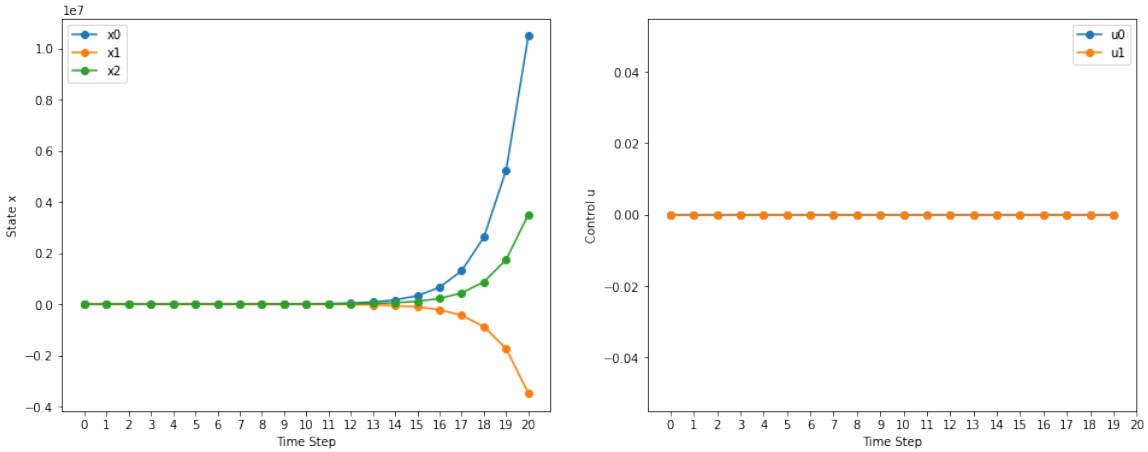

Figure 5: Uncontrolled System 3

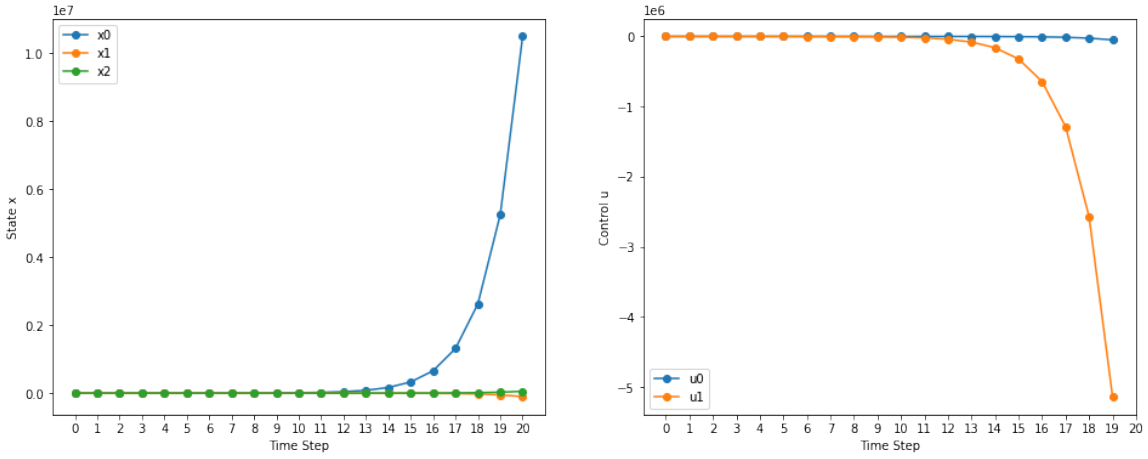

Figure 6: Controlled System 3

From the above simulation results, we can find that for the unstable systems (system 1 and system 3) the state will diverge as time goes by, but for the stable system (system 2) the state will converge to the origin.

For the unstable but controllable system (system 1), LQR could drive the state to the origin with enough time steps.

For the stable and controllable system (system 2), LQR could accelerate the convergence of the system.

For the unstable and uncontrollable system (system 3), LQR could not drive the state to the origin in this case, and the state will diverge even if we have the optimal control policy.

Exercise 2

Question 1

Solution 1) The system should be stabilized around the resting position $\bar{z} = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}^T$. So the cost function at n step can be written as:

$$J_n(x_n) = (z_n - \bar{z})^T (z_n - \bar{z}) + 0.01 u_n^T u_n \tag{11}$$

And the total cost function is the sum of J_n:

$$\sum_{n=0}^{N-1} J_n(x_n) = \sum_{n=0}^{N-1} (z_n - \bar{z})^T (z_n - \bar{z}) + 0.01 u_n^T u_n \tag{12}$$

2) Since the system should be stabilized around the resting position $\bar{z}$, it's a tracking problem where the tracking object is $\bar{z}$ for every time step. To solve this problem, we need to solve the Riccati equation iteratively:

$$K_n = -(R_n + B_n^T P_{n+1} B_n)^{-1} B_n^T P_{n+1} A_n \tag{13}$$

$$P_n = Q_n + A_n^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n \tag{14}$$

$$k_n = -(R_n + B_n^T P_{n+1} B_n)^{-1} B_n^T p_{n+1} \tag{15}$$

$$p_n = q_n + A_n^T p_{n+1} + A_n^T P_{n+1} B_n k_n \tag{16}$$

where $P_N = Q_N$, $p_N = q_N$.
The python code to compute the optimal control policy is shown here:

```
def LQRTracking(A, B, Q, R, N, Xs):
    """
    Solve the LQR tracking problem with given system and cost function.

    Args:
        A: the system dynamics;
        B: the control matrix;
        Q: the quadratic cost matrix, x.T Q x;
        R: the quadratic cost matrix, u.T R u;
        N: the number of time step;
        Xs: the tracking target.

    Return:
        Ks: the linear feedback gains.
        ks: the feed-forward gains.
    """

    def DARE(P, p, q):
        """
        A helper function to solve discrete-time algebraic Riccati equation (DARE)
        for tracking problem for each time step.
        """

        K = -np.linalg.inv(B.T @ P @ B + R) @ B.T @ P @ A
        P = Q + A.T @ P @ A + A.T @ P @ B @ K

        k = -np.linalg.inv(B.T @ P @ B + R) @ B.T @ p
        p = q + A.T @ p  + A.T @ P @ B @ k
```

```
            return K, P, k, p

        Ks = []
        ks = []

        P = Q
        p = -Q @ Xs[-1]

        for t in range(N-1, -1, -1):
            q = -Q @ Xs[t]
            K, P, k, p = DARE(P, p, q)

            Ks = [K] + Ks
            ks = [k] + ks

         return Ks, ks

```

Listing 2: LQR Tracking Algorithm

In this case, the parameters are:

$$A_n = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & \Delta t \frac{m_p g}{m_c} & 1 & 0 \\ 0 & \Delta t \frac{(m_p + m_c) g}{l m_c} & 0 & 1 \end{bmatrix} \tag{17}$$

$$B_n = \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t}{m_c} \\ \frac{\Delta t}{l m_c} \end{bmatrix} \tag{18}$$

$$Q_n = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$R_n = 0.01 \tag{20}$$

$$q_n = -Q_n \bar{z} \tag{21}$$

And the optimal control policy is:

$$u_n = K_n(z_n - \bar{z}) + k_n \tag{22}$$

If we solve the equations for a long enough time, K_n and k_n will converge and these are the infinite horizon gains we want:

$$K = \begin{bmatrix} 9.53146 & -220.97185 & 20.19384 & -67.99139 \end{bmatrix} \tag{23}$$

$$k = -0.00005 \tag{24}$$

3) We can use the cost function and algorithm in the previous questions to solve this one. If we start the simulation at $z_0 = \begin{bmatrix} 0.5 & \pi + 0.3 & 0 & 0 \end{bmatrix}^T$, the time evolution of the control input and the states of the system are shown below:

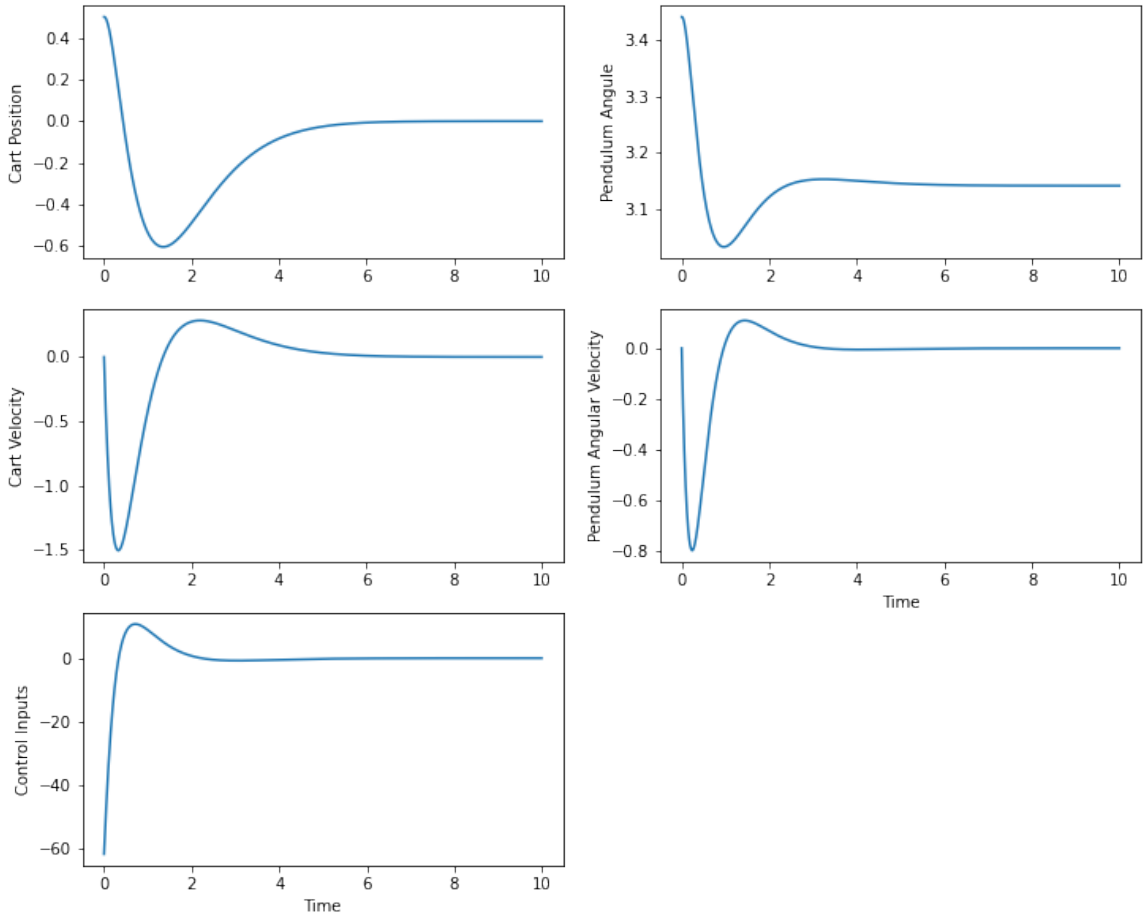

Figure 7: Time Evolution of the States and Controls

The animation can be found here. As we see, the system stabilizes around the resting position.

4) If we run the same simulation but with a different initial state $z_0 = \begin{bmatrix} 0.5 & 0.3 & 0 & 0 \end{bmatrix}^T$, the simulation results are shown below:

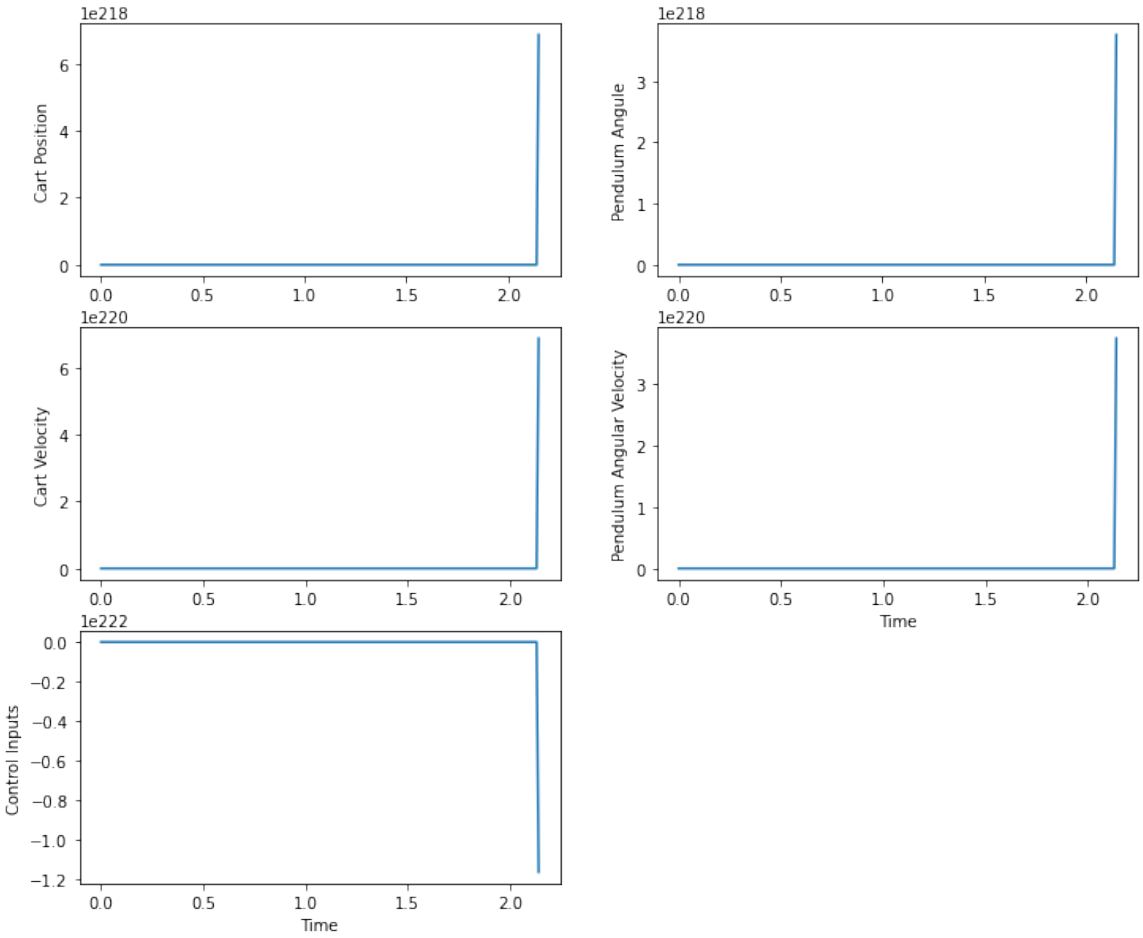

Figure 8: Time Evolution of the States and Controls

It seems that the system diverge from the initial state and our optimal control policy cannot stabilize the system. The reason why this happens is that our system is a non-linear system and we only linearize it around the resting position $\bar{z} = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}^T$. In this case the initial state z_0 is too far from the resting position and the linearization is not valid. So our optimal control policy is not useful in this case.

Question 2

Solution 1) In this question, the cart position is defined as:

$$x(t) = \sin(2\pi f t) \tag{25}$$

where $f = 0.5\,\text{Hz}$ is the frequency. And the corresponding cart velocity is:

$$v(t) = \dot{x}(t) = 2\pi f \cos(2\pi f t) \tag{26}$$

Therefore, the tracking trajectory for each time step is:

$$\bar{z}_n = \begin{bmatrix} x(n\Delta t) & \pi & v(n\Delta t) & 0 \end{bmatrix}^T \tag{27}$$

The cost function in this question is almost the same as the function in the previous one, with the only difference that the tracking trajectory changes over time:

$$\sum_{n=0}^{N-1} J_n(x_n) = \sum_{n=0}^{N-1} (z_n - \bar{z}_n)^T (z_n - \bar{z}_n) + 0.01 u_n^T u_n \tag{28}$$

2) This question is the same as Question 1, We can use the codes in Listing 2 to get the sequence of controller. More details can be found in the jupyter-notebook file. The controller for each time step is:

$$u_t = K_t(z_t - \bar{z}_t) + k_t \tag{29}$$

where u_t is the control input at time t; K_t and k_t are the control gains; z_t is the state at time t; $\bar{z}_t$ is the tracking trajectory at time t.

3) If we run the simulation with the above cost function and controller for $N = 1000$ steps, we'll get the simulation below:

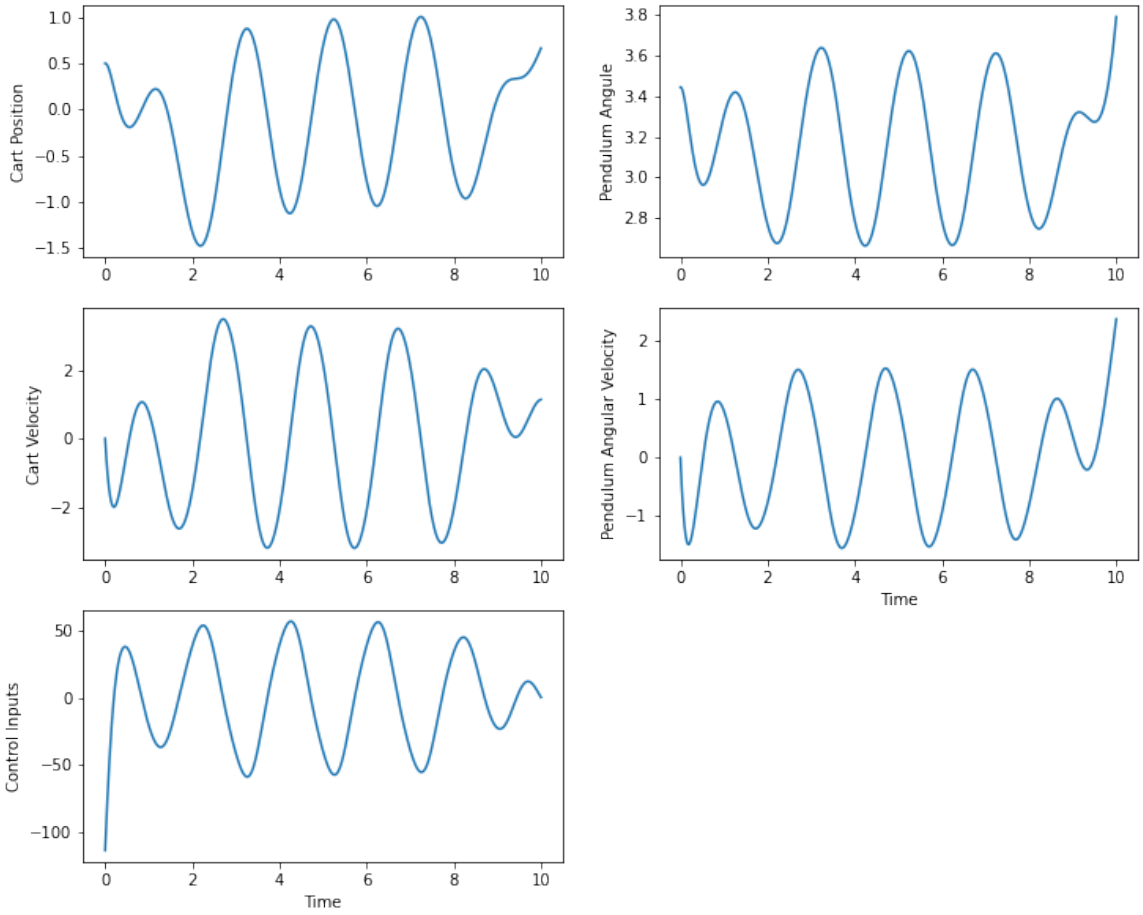

Figure 9: Time Evolution of the States and Controls ($N = 1000$)

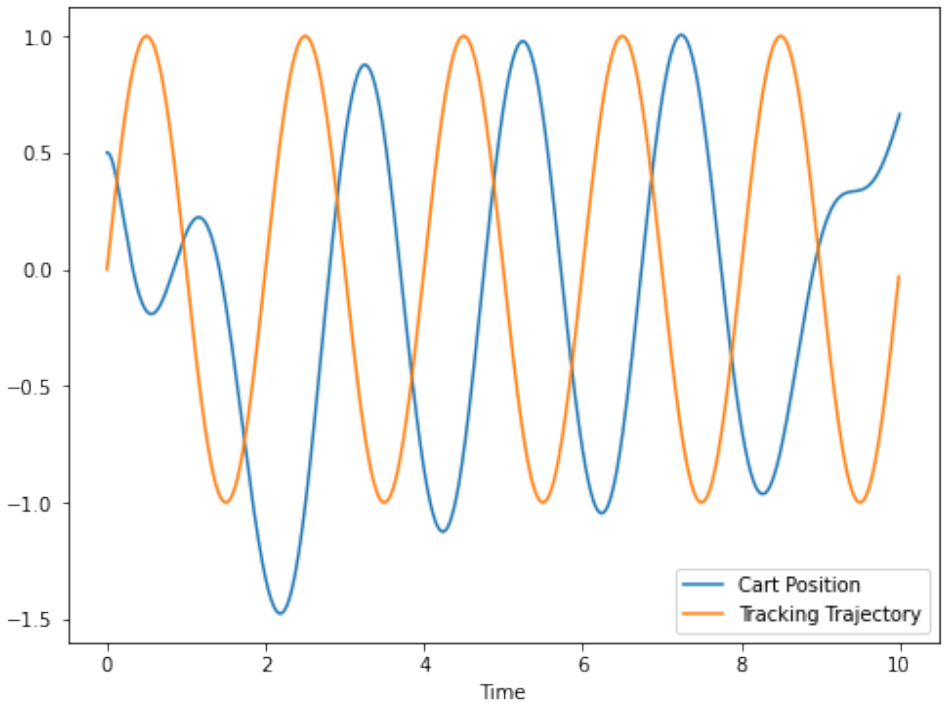

Figure 10: Cart Position and Tracking Trajectory ($N = 1000$)

The animation can be found here. As we see, the controller does dirve the cart to follow a sine trajectory.

If we run the simulation for a longer time (e.g. $N = 1100$), the system will have a better performance at the ending steps. The animation can be found here, and the simulation results are shown below:

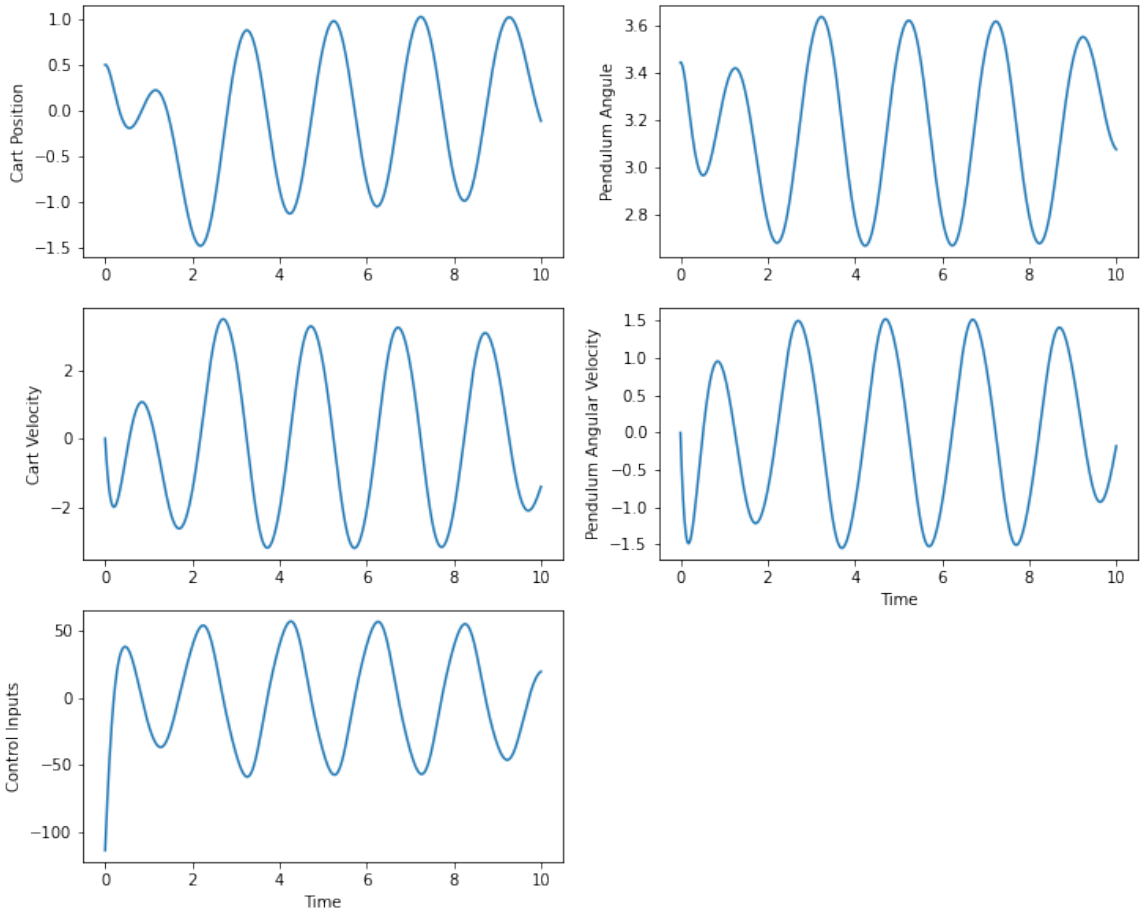

Figure 11: Time Evolution of the States and Controls ($N = 1100$)

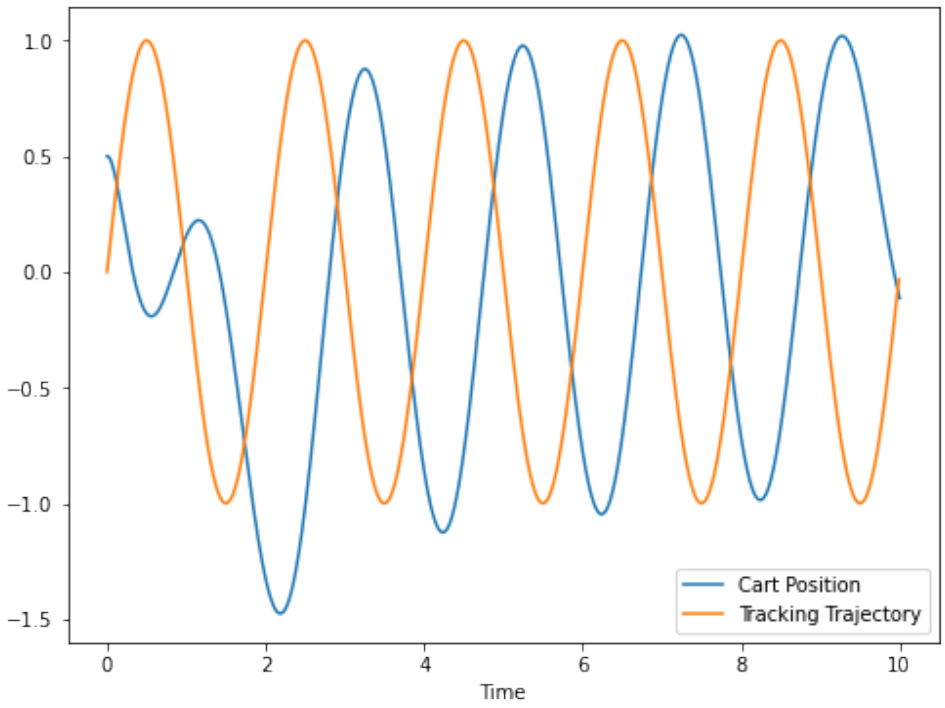

Figure 12: Cart Position and Tracking Trajectory ($N = 1100$)

As we see, the controller could drive the cart following the sine trajectory with only a phase difference.

4) The cost value predicted by the controller is:

$$J_n(x_n) = x_n^T P_n x_n + 2p_n^T x_n + C \tag{30}$$

where P_n and p_n are the matrix defined in Ricatti equation, $P_n = Q_n + A^T P_{n+1} A + A^T P_{n+1} B K_n$, $p_n = q_n + A^T p_{n+1} + A^T P_{n+1} B k_n$; x_n is the real state at time step n; and C is a constant. Note this predicted cost is an estimation of "cost-to-go" from the current time step n.

The predicted cost and real cost for this cart-pole tracking problem is shown below:

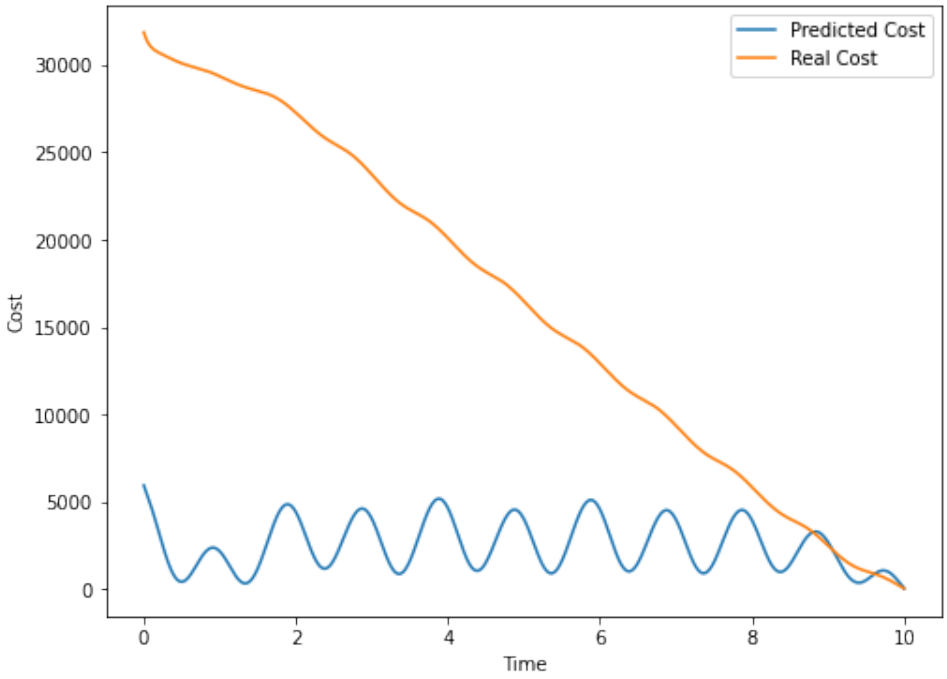

Figure 13: Predicted Cost and Real Cost

It seems that the predicted cost is much lower than the real one, one of the reasons is that we are omitting the constant part C in predicting the cost function. The constant C comes from the tracking target $\bar{z}_t$ and accumulates through time, so this may lead to a huge difference when we recur backwards. Another reason is that we are only using a linearized dynamic and control matrix, and it may not be suitable when the system is far from the origin (where we linearize the system). In this case, the control input may not be optimal for the current state and the predicted cost is therefore not a proper estimation.