Exercise 1 :

1) $$\mathbf{x}_{n+1} = \begin{bmatrix} 0.5 & 0. & 0.5 \\ 0. & 0. & -2. \\ 4. & 2. & 1. \end{bmatrix} \mathbf{x}_n + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_n$$

1a) $A=\begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & -2 \\ 4 & 2 & 1 \end{bmatrix}$ $B=\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

When $u_n = 0$,

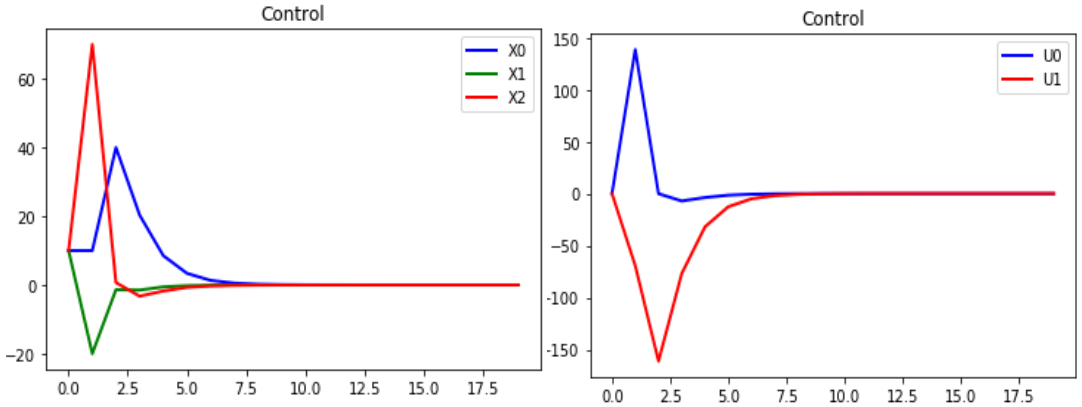
Eigenvalue of A: $\lambda_1 = 1$ $\lambda_2 = 0.25 + 1.3919i$ $\lambda_3 = 0.25 - 1.3919i$

Because the norm of eigenvalue is bigger than 1 => $|\lambda| > 1$, system is unstable.

1b) Co= [B AB A^2B]$=\begin{bmatrix} 0 & 0 & 0 & 0.5 & 1 & 0.75 \\ 1 & 0 & 0 & -2 & -4 & -2 \\ 0 & 1 & 2 & 1 & 2 & -1 \end{bmatrix}$

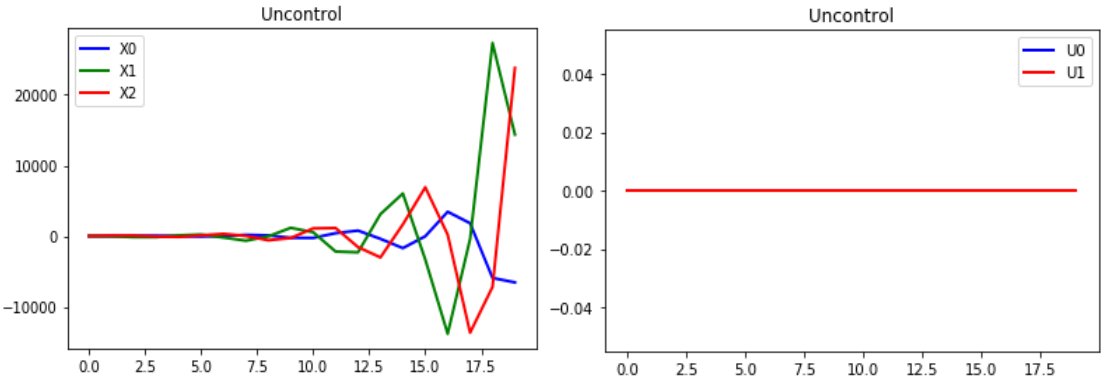
Rank(Co)=3, n=3 => Co is full rank, system is controllable

1d) Controlled system:



X U

Uncontrolled system:



X U

From the results, we can see that the uncontrolled system is not stable, but the system is controllable, which is same as the conclusion in previous questions. LQR could drive the state to the origin with enough time step.

Optimal gains: (every K is 2x3matrix)

```
K [[ 0.       0.       0.       0.       0.       1.9802  -0.03623 -0.00029  1.94514 -0.03662 -0.0003   1.94477 -0.03674
  -0.00031  1.94466 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031
   1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464
  -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676
  -0.00031  1.94464 -0.03676 -0.00031  1.94464 -0.03676 -0.00031  1.94464]
 [ 0.       0.       0.      -3.9604  -1.9802  -0.9901  -4.07948 -1.98472 -1.10211 -4.10667 -1.98566 -1.12788 -4.11089
  -1.9858  -1.13188 -4.11154 -1.98583 -1.1325  -4.11164 -1.98583 -1.13259 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583
  -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261
  -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166
  -1.98583 -1.13261 -4.11166 -1.98583 -1.13261 -4.11166 -1.98583 -1.13261]]
```

2) $$\mathbf{x}_{n+1} = \begin{bmatrix} 0.5 & 0. & 0.5 \\ 0. & 0. & -0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} \mathbf{x}_n + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_n$$

2a) A=$\begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & -0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$ B=$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

When $u_n = 0$,

Eigenvalue of A: $\lambda_1 = 0.8774$ $\lambda_2 = 0.0613 + 0.3724i$

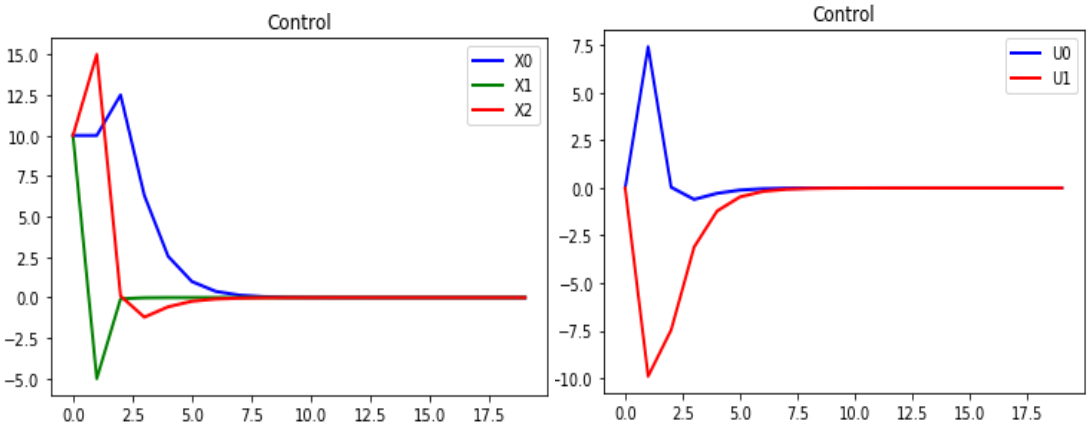
$\lambda_3 = 0.0613 - 0.3724i$

Because the norm of eigenvalue is smaller than 1 => $|\lambda| < 1$, system is stable.

2b) Co= [B AB A^2B]=$\begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.25 & 0.5 \\ 1 & 0 & 0 & -0.5 & -0.25 & -0.25 \\ 0 & 1 & 0.5 & 0.5 & 0.25 & 0.25 \end{bmatrix}$

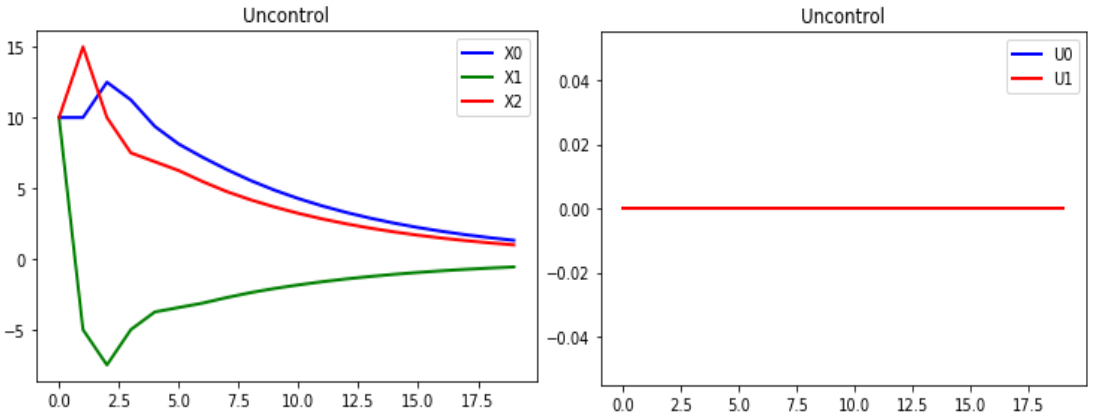
Rank(Co)=3, n=3 => Co is full rank, system is controllable

2d) Controlled system:



X U

Uncontrolled system:



X U

From the results, we can see that the uncontrolled system is stable, and the system is also controllable, which is same as the conclusion in previous questions. LQR could accelerate the convergence of the system.

Optimal gains: (every K is 2x3matrix)

```
K [[ 0.       0.       0.       0.       0.       0.49505 -0.00099 -0.00001  0.49407 -0.00114 -0.00001  0.49392 -0.00117
  -0.00001  0.4939  -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001
   0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389
  -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117
  -0.00001  0.49389 -0.00117 -0.00001  0.49389 -0.00117 -0.00001  0.49389]
 [ 0.       0.       0.      -0.49505 -0.49505 -0.49505 -0.59584 -0.49605 -0.59583 -0.61166 -0.4962  -0.61165 -0.61403
  -0.49623 -0.61402 -0.61438 -0.49623 -0.61437 -0.61443 -0.49623 -0.61442 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623
  -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443
  -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444
  -0.49623 -0.61443 -0.61444 -0.49623 -0.61443 -0.61444 -0.49623 -0.61443]]
```

3) $$\mathbf{x}_{n+1} = \begin{bmatrix} 0.5 & 0. & 0. \\ 0. & 0. & -2. \\ 1. & 1. & 0. \end{bmatrix} \mathbf{x}_n + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}_n$$

3a) $A=\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0 & -2 \\ 1 & 1 & 0 \end{bmatrix}$ $B=\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$

When $u_n = 0$,

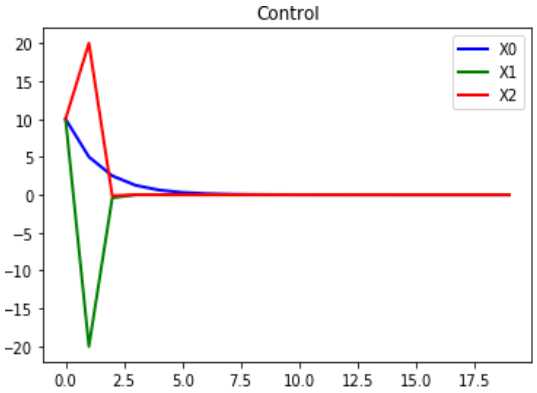
Eigenvalue of A: $\lambda_1 = 1.4142i$ $\lambda_2 = -1.4142i$ $\lambda_3 = 0.5$

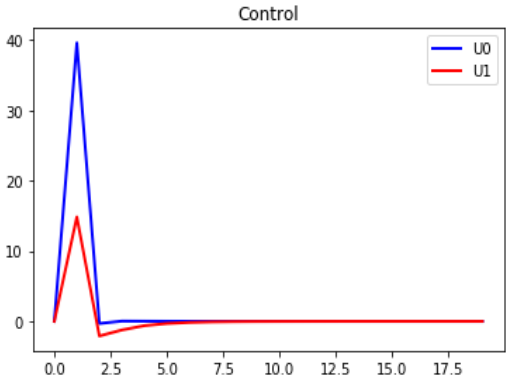
Because the norm of eigenvalue is bigger than 1 => $|\lambda| > 1$, system is stable.

3b) Co= [B AB A^2B]$=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 & -2 & 0 \\ 0 & 1 & 1 & 0 & 0 & -2 \end{bmatrix}$

Rank(Co)=2, n=3 => Co is not full rank, system is uncontrollable

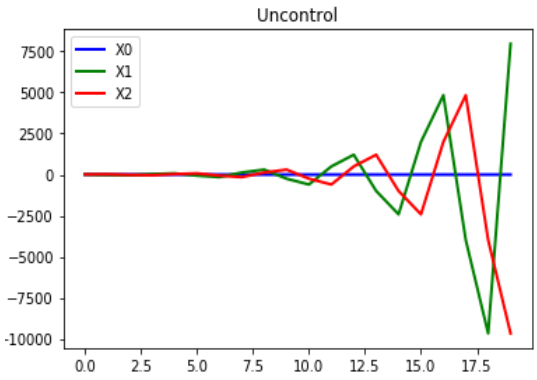
3d) Controlled system:

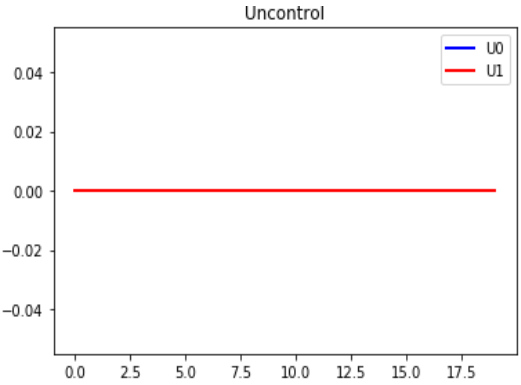




X U

Uncontrolled system:





X U

From the results, we can see that the uncontrolled system 3 is not stable, and the system is not controllable as well, which is same as the conclusion in previous questions. 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

Optimal gains: (every K is 2x3matrix)

```
K [[ 0.       0.       0.       0.       0.       1.9802  -0.00485  0.       1.98039 -0.00486  0.       1.98039 -0.00486
   0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.
   1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039
  -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039 -0.00486
   0.       1.98039 -0.00486  0.       1.98039 -0.00486  0.       1.98039]
 [ 0.       0.       0.      -0.9901  -0.9901   0.      -0.99047 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043
  -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047
   0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.
  -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.      -0.99043
  -0.99047  0.      -0.99043 -0.99047  0.      -0.99043 -0.99047  0.     ]]
```

1-3c) As long as the dynamic system is controllable, we can expect a LQR design to be able to drive the system to the origin from any initial conditions. Besides, LQR could make the system stable for infinity horizons. Thus, for system 1 and system 2, there is possible LQR design to be able to drive the system to the origin from any initial conditions for infinity horizons. But for system 3 there is no guarantee of this.

Exercise 2:

Q1:

1)

$$\dot{x} = v$$

$$\dot{\theta} = \omega$$

$$\dot{v} = \frac{u + m_p sin\theta(l\omega^2 + gcos\theta)}{m_c + m_p \sin^2\theta}$$

$$\dot{\omega} = \frac{-ucos\theta - m_p l\omega^2 cos\theta sin\theta - (m_c + m_p)gsin\theta)}{m_c + m_p \sin^2\theta}$$

$z = [x\ \theta\ v\ \omega]^T$ target position: $\bar{z} = [0\ \ \pi\ 0\ 0]^T$ $\tilde{z} = z - \bar{z}$

$\dot{z} = [\dot{x}\ \ \dot{\theta}\ \ \dot{v}\ \ \dot{\omega}]^T$

The linearized dynamics equation is:

$$\tilde{z}_{n+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & \frac{\Delta t m_p g}{m_c} & 1 & 0 \\ 0 & \frac{\Delta t(m_p + m_c)g}{l m_c} & 0 & 1 \end{bmatrix} \tilde{z}_n + \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t}{m_c} \\ \frac{\Delta t}{l m_c} \end{bmatrix} \tilde{u}_n$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & \frac{\Delta t m_p g}{m_c} & 1 & 0 \\ 0 & \frac{\Delta t(m_p + m_c)g}{l m_c} & 0 & 1 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t}{m_c} \\ \frac{\Delta t}{l m_c} \end{bmatrix}$$

After several tries: set Q=1 R=0.01 （Q is 4x4 unit matrix）

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ R = 0.01$$

Cost function: $\tilde{z} = z_n - \bar{z}$

$J_N(x_N) = \tilde{z}^{\bar{T}} Q \tilde{z} + \sum_{n=0}^{N-1} \tilde{z}^{\bar{T}} Q \tilde{z} + \tilde{u}^{\bar{T}} R \tilde{u}$ for total cost

$J_n(x_n) = \tilde{z}^{\bar{T}} Q \tilde{z} + \tilde{u}^{\bar{T}} R \tilde{u}$ for every step cost

2) $x_{n+1} = Ax_n + Bu_n$ $u_n = K_n(z_n - \bar{z}) + k_n$

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

$$P_n = Q_n + A^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n$$

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

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

$$P_N = Q_N, p_N = q_N \quad q_n = -Q_n \bar{z}$$

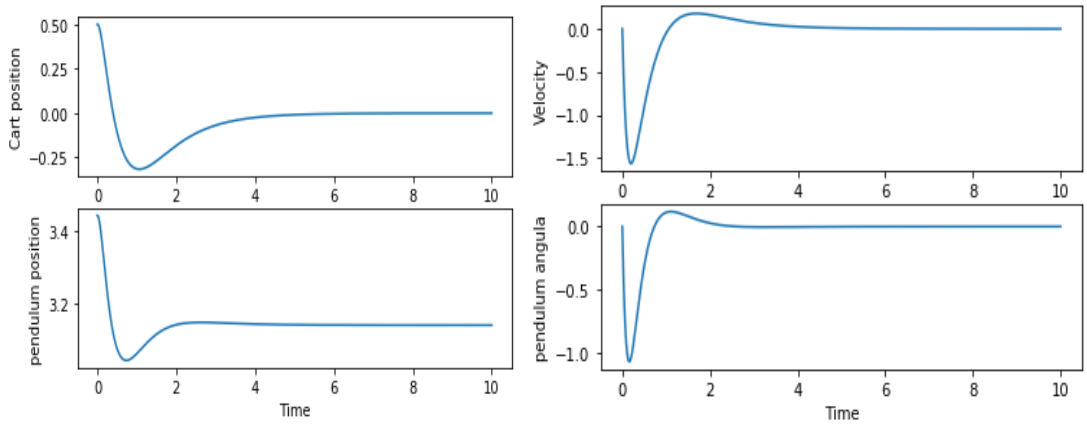
According to cost function and equations above, we can calculate every gain

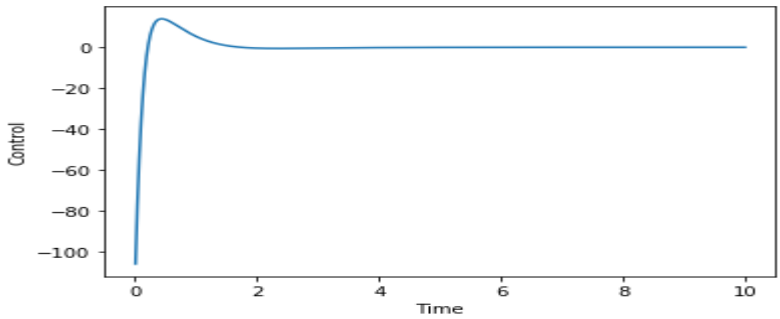
3)codes of function for getting control and Riccati parameter:

```
def solve_ricatti_equations(A,B,Q,R,horizon_length):

    P = [] #will contain the list of Ps from N to 0
    K = [] #will contain the list of Ks from N-1 to 0
    k = []
    p = []
    P.append(Q) #PN
    p.append(-1.0*Q.dot(np.array([0.,np.pi,0.,0.])))
    for i in range(horizon_length):
        Knew = -1.0 * np.linalg.inv(B.transpose().dot(P[i]).dot(B) + R).dot(B.transpose()).dot(P[i]).dot(A)
        Pnew = Q + A.transpose().dot(P[i]).dot(A) + A.transpose().dot(P[i]).dot(B).dot(Knew)
        knew = -1.0 * np.linalg.inv(B.transpose().dot(P[i]).dot(B) + R).dot(B.transpose()).dot(p[i])
        pnew = -1.0 * Q.dot(np.array([0.,np.pi,0.,0.])) + A.transpose().dot(p[i]) + A.transpose().dot(P[i]).dot(B).dot(knew)
        K.append(Knew)
        P.append(Pnew)
        k.append(knew)
        p.append(pnew)

    # since we went backward we return reverted lists
    return P[::-1],K[::-1],p[::-1],k[::-1]
def controller(z,i,K,k):
    u = K.dot(z)- K.dot(np.array([0.,np.pi,0.,0.]))

    return u
```

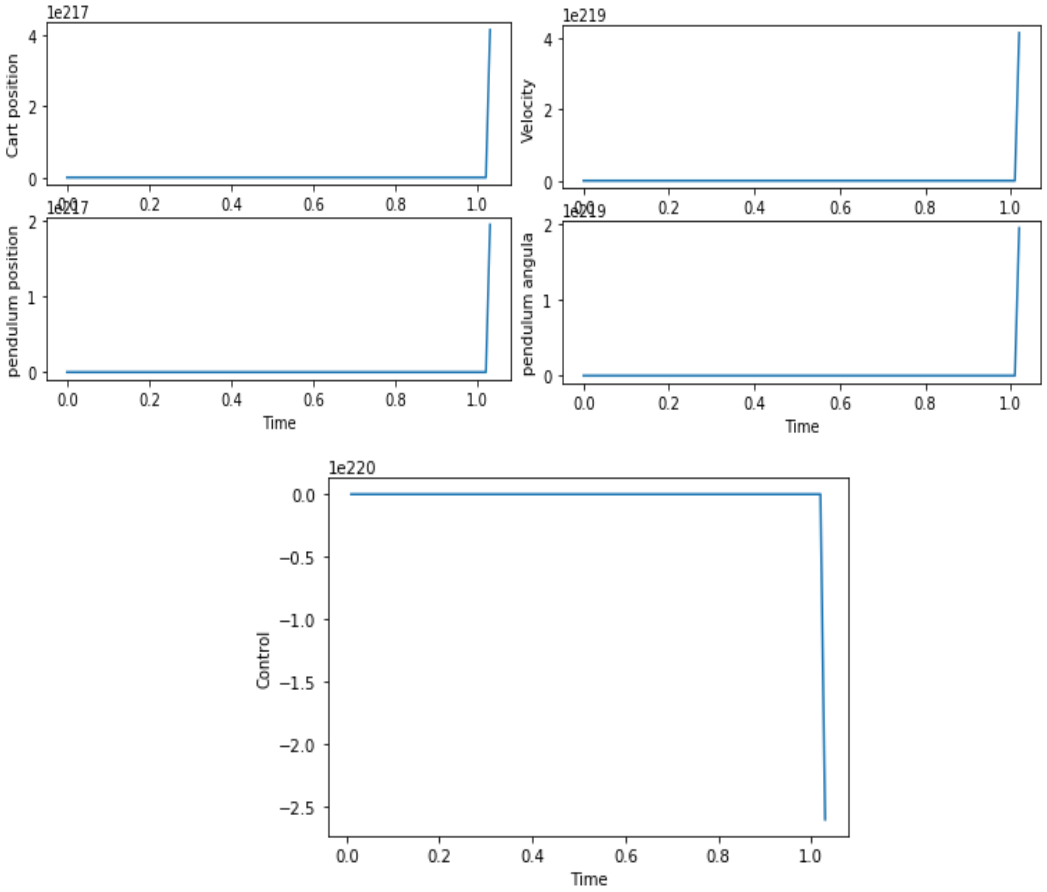
Initial point: $z_0 = [0.5,\ \pi + 0.3,\ 0.,\ 0.]^T$.

Result of z:





4) Initial point: $z_0 = [0.5,\ 0.3,\ 0.,\ 0.]^T$



In this situation, the controller doesn't work well. Because the distance 0.3 to pi is too big for this control system. It cannot properly control the system to the fixed point we want. Thus, it shows sudden change on graph.

Q2:

1)

$$\dot{x} = v$$

$$\dot{\theta} = \omega$$

$$\dot{v} = \frac{u + m_p sin\theta(l\omega^2 + gcos\theta)}{m_c + m_p \sin^2\theta}$$

$$\dot{\omega} = \frac{-ucos\theta - m_p l\omega^2 cos\theta sin\theta - (m_c + m_p)gsin\theta)}{m_c + m_p \sin^2\theta}$$

$z = [x\ \theta\ v\ \omega]^T$
$\dot{z} = [\dot{x}\ \ \dot{\theta}\ \ \dot{v}\ \ \dot{\omega}]^T$
target: $\bar{z} = [\bar{x}\ \ \bar{\theta}\ \bar{v}\ \bar{\omega}\ ]^T$
$\tilde{z} = z - \bar{z}$

$\bar{x} = \sin(\pi t)$ $\dot{\bar{x}} = \bar{v} = \pi\cos(\pi t)$
target trajectory: $\bar{z} = [\sin(\pi t)\ \ \pi\ \ \pi\cos(\pi t)\ \ 0]^T$

The linearized dynamics equation is:

$$\tilde{z}_{n+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & \frac{\Delta t m_p g}{m_c} & 1 & 0 \\ 0 & \frac{\Delta t(m_p + m_c)g}{l m_c} & 0 & 1 \end{bmatrix} \tilde{z}_n + \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t}{m_c} \\ \frac{\Delta t}{l m_c} \end{bmatrix} \tilde{u}_n$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & \frac{\Delta t m_p g}{m_c} & 1 & 0 \\ 0 & \frac{\Delta t(m_p+m_c)g}{l m_c} & 0 & 1 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t}{m_c} \\ \frac{\Delta t}{l m_c} \end{bmatrix}$$

After several tries: set Q=1 R=0.01 （Q is 4x4 unit matrix）

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ R = 0.01$$

Cost function: $\tilde{z} = z_n - \bar{z}$
$J_N(x_N) = \tilde{z}^{\bar{T}} Q\tilde{z} + \sum_{n=0}^{N-1} \tilde{z}^{\bar{T}} Q\tilde{z} + \tilde{u}^{\bar{T}} R\tilde{u}$ for total cost
$J_n(x_n) = \tilde{z}^{\bar{T}} Q\tilde{z} + \tilde{u}^{\bar{T}} R\tilde{u}$ for every step cost

2)

$$x_{n+1} = Ax_n + Bu_n \qquad u_n = K_n(z_n - \bar{z}) + k_n$$

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

$$P_n = Q_n + A^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n$$

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

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

$$P_N = Q_N, p_N = q_N \quad q_n = -Q_n \bar{z}$$

According to cost function from (1) and equations above, we can calculate every gain

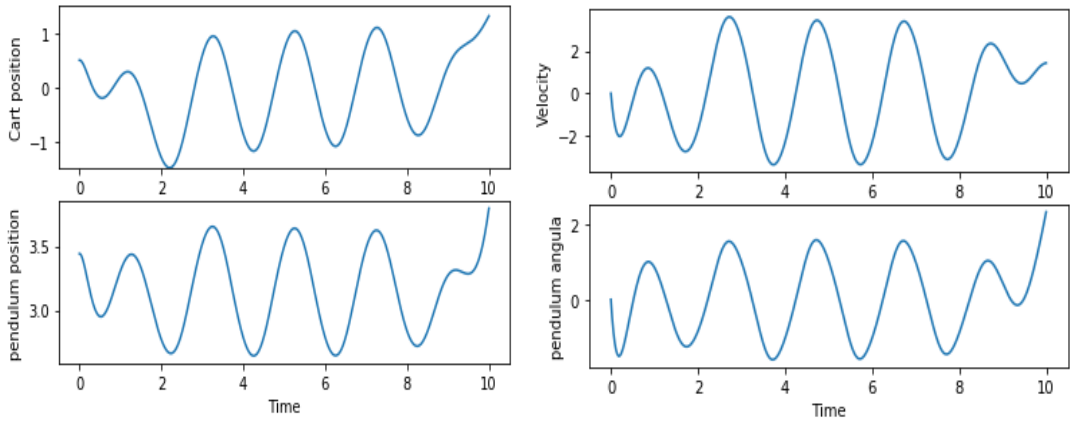
3) codes of function for getting control and Riccati parameter:

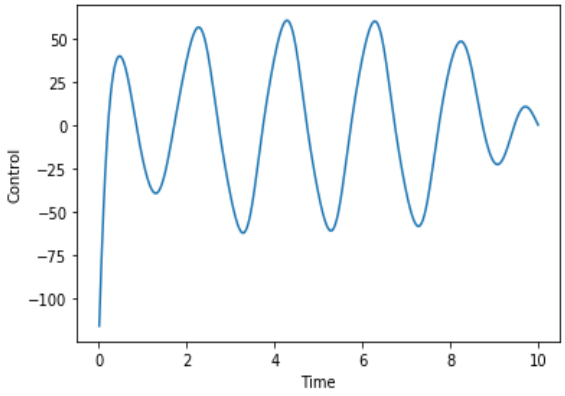
```
def solve_ricatti_equations(A,B,Q,R,Z,horizon_length):

    P = [] #will contain the list of Ps from N to 0
    K = [] #will contain the list of Ks from N-1 to 0
    k = []
    p = []
    P.append(Q) #Pn inital
    p.append(-1.0*Q.dot(Z[:,0])) #pn initial
    for i in range(horizon_length):
        Knew = -1.0 * np.linalg.inv(B.transpose().dot(P[i]).dot(B) + R).dot(B.transpose()).dot(P[i]).dot(A)
        Pnew = Q + A.transpose().dot(P[i]).dot(A) + A.transpose().dot(P[i]).dot(B).dot(Knew)
        knew = -1.0 * np.linalg.inv(B.transpose().dot(P[i]).dot(B) + R).dot(B.transpose()).dot(p[i])
        pnew = -1.0 * Q.dot(Z[:,i]) + A.transpose().dot(p[i]) + A.transpose().dot(P[i]).dot(B).dot(knew)
        K.append(Knew)
        P.append(Pnew)
        k.append(knew)
        p.append(pnew)

    # since we went backward we return reverted lists
    return P[::-1],K[::-1],p[::-1],k[::-1]
def controller(z, i, K, k, Z):
    u = K.dot(z)- K.dot(Z) + k
    return u
```

In this situation, it's tracking problem. We need to think about k, p

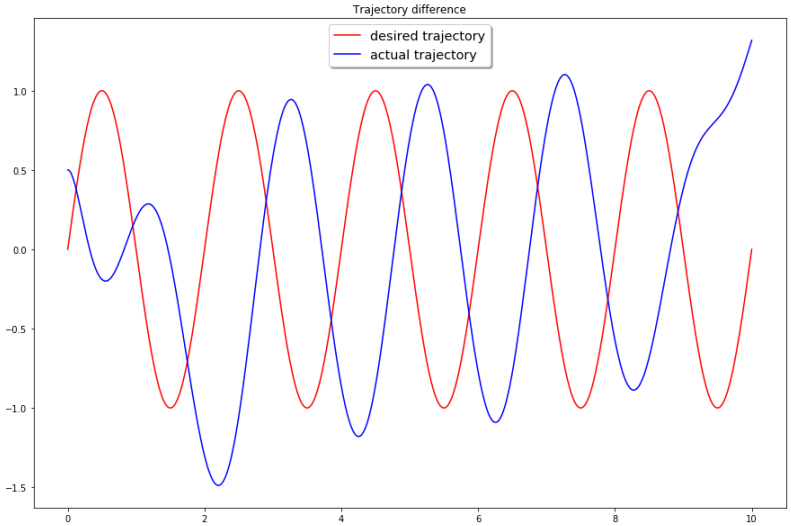
Initial condition: $z_0 = [0.5,\ \pi + 0.3,\ 0.,\ 0.]^T$.

Result of Z:



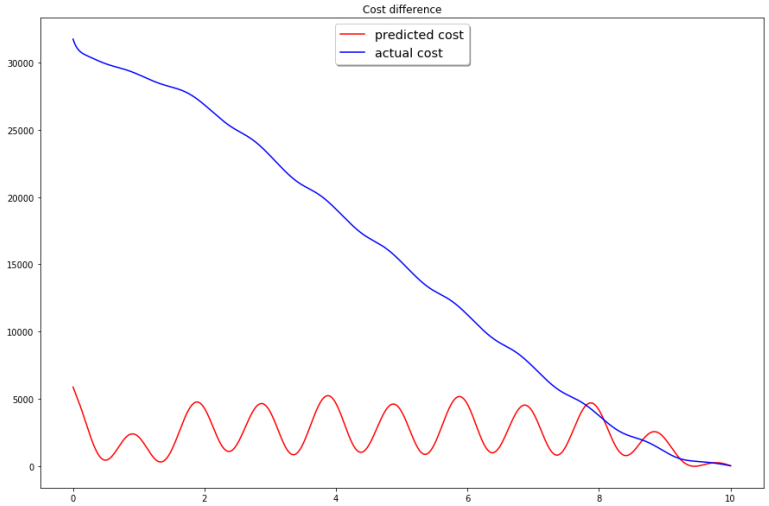


Comparison between control and desired trajectory:



According to the graph above, the actual trajectory tracks well in the middle period. But there is still displacement at the beginning and the end of actual trajectory.

4) Comparison between control and desired cost:



According to the graph above, it is obvious that predicted cost is much lower than actual cost and actual cost decreases. Because when we linearize the nonlinear system, we usually use Taylor's expansion to approximately calculate next step of $z = [x\ \theta\ v\ \omega]^T$. In every step, it is a proximate value. After the 1000 times backward iteration, the error would become larger, and cost would accumulate. Thus, actual cost is getting smaller and smaller.