Homework 01

AA203: Optimal and learning-based Control

Ricardo Luiz Moreira Paschoeto

rp304154@stanford.edu

Problem 1: Direct Model-reference Adaptive Control (MRAC),

$$\dot{y}(t) + \alpha y(t) = \beta u(t) \text{ (I)}$$

$$\dot{y_m}(t) + \alpha_m y_m(t) = \beta_m r(t)\ (II)$$

a) Control law:

$$u(t) = k_r(t)r(t) + k_y(t)y(t), \text{ (III)}$$

$$y(0) = y_m(0), \text{ (IV)}$$

$$k_r^* = \frac{\beta_m}{\beta}\ (V), k_y^* = \frac{\alpha - \alpha_m}{\beta} \text{ (VI)}.$$

Substituting (III) in (I),

$$\dot{y}(t) + \alpha y(t) = \beta(k_r(t)r(t) + k_y(t)y(t))$$

$$\dot{y}(t) + \alpha y(t) = \beta k_r(t)r(t) + \beta k_y(t)y(t)$$

$$\dot{y}(t) + y(t)\left(\alpha - \beta k_y^*(t)\right) = \beta k_r^*(t)r(t)$$

Using (V) and (VI),

$$\dot{y}(t) + y(t)\left(\alpha - \beta\left(\frac{\alpha - \alpha_m}{\beta}\right)\right) = \beta\frac{\beta_m}{\beta}r(t)$$

Finally True plant match the reference model,

$$\dot{y}(t) + \alpha_m y(t) = \beta_m r(t),$$

b) Given,

$$e(t) = y(t) - y_m(t),$$

$$\delta_r(t) = k_r(t) - k_r^*,$$

$$\delta_y(t) = k_y(t) - k_y^*.$$

Find Differential equation for *e*,

$$\dot{e}(t) = \dot{y}(t) - \dot{y_m}(t),$$

$$\dot{e}(t) = \beta u(t) - \alpha y(t) - \beta_m r(t) + \alpha_m y_m(t),$$

$$\dot{e}(t) = \beta k_r(t)r(t) + \beta k_y(t)y(t) - \alpha y(t) - \beta_m r(t) + \alpha_m y_m(t),$$

$$\dot{e}(t) = \beta\left(\delta_r(t) + \frac{\beta_m}{\beta}\right)r(t) + \beta\left(\delta_y(t) + \frac{\alpha - \alpha_m}{\beta}\right)y(t) - \alpha y(t) - \beta_m r(t) + \alpha_m y_m(t),$$

Manipulating the equation above,

$$\dot{e}(t) - \beta(\delta_r(t)r(t) + \delta_y(t)y(t)) + \alpha_m e(t) = 0,$$

$$\dot{e}(t) + \alpha_m e(t) = \beta(\delta_r(t)r(t) + \delta_y(t)y(t)) \text{ (VII)}.$$

c) Show that $\dot{V} = -\alpha_m e^2$, given $V(x) = \frac{1}{2}e^2 + \frac{|\beta|}{2\gamma}(\delta_r^2 + \delta_y^2)$.

$$\dot{V} = e\dot{e} + \frac{|\beta|}{\gamma}(\delta_r\dot{\delta_r} + \delta_y\dot{\delta_y}), \text{ (VIII)}$$

Substituting (VII) in (VIII), and $\dot{\delta_r} = \dot{k_r}$, $\dot{\delta_y} = \dot{k_y}$.

$$\dot{V} = e\left(\left(\beta(\delta_r(t)r(t) + \delta_y(t)y(t)\right) - \alpha_m e(t)\right) + \frac{|\beta|}{2\gamma}(\delta_r\dot{k}_r + \delta_y\dot{k}_y),$$

Using the adaptation law,

$$\dot{k_r}(t) = -sign(\beta)\gamma e(t)r(t),$$

$$\dot{k_y}(t) = -sign(\beta)\gamma e(t)y(t),$$

We have,

$$\dot{V} = (\beta - |\beta|sign(\beta))r(t)\delta_r r(t) + (\beta - |\beta|sign(\beta))e(t)\delta_y y - \alpha_m e^2,$$

Using the property,

$$x = |x|sign(x)$$

We have,

$$\dot{V} = -\alpha_m e^2.$$

Where $e^2 \geq 0$, $\delta_r^2 > 0$ and $\delta_y^2 > 0$, so V is definite-positive in x for $t \geq 0$. $\dot{V}$ is a function only of e^2 and $\dot{V}$ is negative semi-definite, *e(t)* is bounded.

d) Barbalat's Lemma

$$\dot{V} = -\alpha_m e^2,$$

We have *r(t)* bounded, so $y_m(t), u(t) and\ y(t)$ are bounded, therefore, *e(t)* is bounded, $e(t) = y(t) - y_m(t)$, and the function $\dot{V}$ is bounded, then from Barbalat's Lemma $\lim_{t\to\infty} \dot{V}(t) = 0$. V is differentiable and has bounded derivate, so is uniformly continuous.

e) Apply MRAC to unstable plant,

$$\dot{y}(t) - y(t) = 3u(t).$$

Reference model,

$$\dot{y_m}(t) + 4y_m(t) = 4r(t).$$

Initial conditions,

$$y(0) = 0,$$

$$e(0) = 0,$$

$$k_r(0) = 0,$$

$$k_y(0) = 0,$$

$$\delta_r(0) = -k_r^*,$$

$$\delta_y(0) = -k_y^*.$$

Equations,

$$\dot{y}(t) = \left(1 + 3k_y(t)\right)y(t) + 3k_r(t)r(t),$$

$$\dot{e}(t) = y(t) - y_m(t),$$

$$\dot{k}_r(t) = -sign(\beta)\gamma e(t)r(t),$$

$$\dot{k}_y(t) = -sign(\beta)\gamma e(t)y(t),$$

$$\delta_r(t) = k_r(t) - k_r^*,$$

$$\delta_y(t) = k_y(t) - k_y^*.$$

Results,

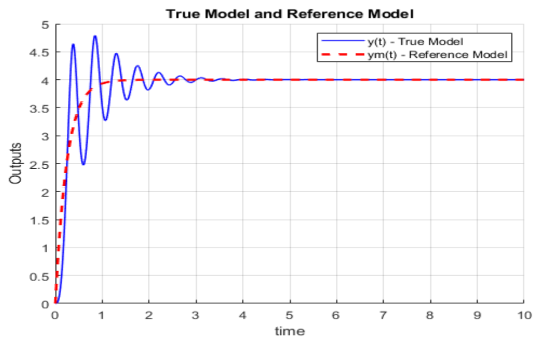

Figure 1

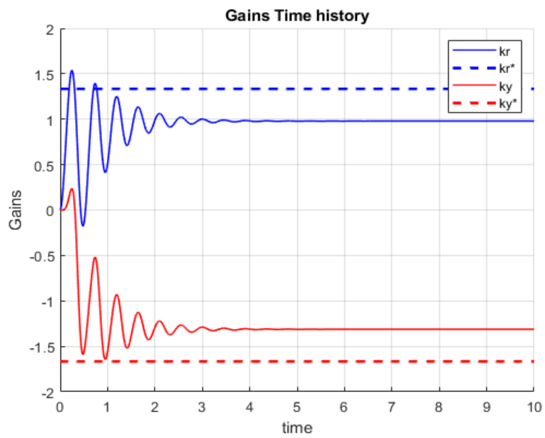

Figure 2

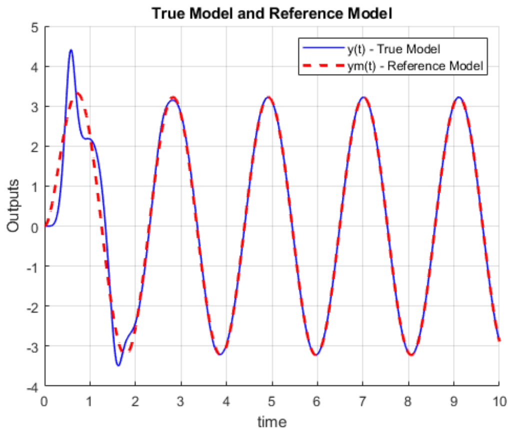

Figure 3

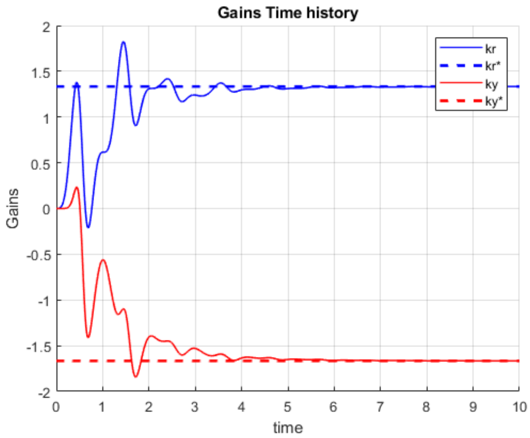

Figure 4

The trend for signals $k_r(t)$ and $k_y(t)$ for *r(t)=4* (figure 2) has an error in steady state. In both simulations the error *e(t)* = $y(t) - y_m(t)$ = 0, and for the *r(t) = 4sin(3t)* the trend for the signals $k_r(t)$and $k_y(t)$ has error zero in steady state, My conclusion is related with the Boundedness of *r(t)* (in the case of *r(t)=4* is unbounded), that has influence in the Boundedness of error functions $\delta_r(t)$ and $\delta_y(t)$ for $k_r(t)$ and $k_y(t)$, respectively, when $t \rightarrow \infty$ (Lyapunov and Barbalat's Lemma).

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt

alpha = -1
beta = 3
alpha_m = 4
beta_m = 4
dt = 0.01
gamma = 2.0
tval = np.linspace(0,10,int((10/dt)))
y = 0
ym = 0
kr = 0
```

```
ky = 0

y_history = []
ym_history = []
kr_history = []
ky_history = []
error_history = []
delta_r_hist = []
delta_y_hist = []

y_history.append(y)
ym_history.append(ym)
kr_history.append(kr)
ky_history.append(ky)

kr_star = beta_m/beta
ky_star = (alpha - alpha_m)/beta
kr_star_hist = kr_star*np.ones(len(tval))
ky_star_hist = ky_star*np.ones(len(tval))

error = 0
error_history.append(error)
dr = 0 - kr_star
delta_r_hist.append(dr)
dy = 0 - ky_star
delta_y_hist.append(dy)

ref_in = 1

def ref(t):
    if ref_in == 0:
        r = 4
    else:
        r = 4*np.sin(3*t)
    return r

def y_dot(y, r, ky, kr):
    return 3*control(y, r, ky, kr) + y

def control(y, r, ky, kr):
    return kr*r + ky*y

def ym_dot(ym, r):
    return 4*(r - ym)

def kr_dot(gamma, error, r):
    return -gamma*error*r

def ky_dot(gamma, error, y):
```

```python
    return -gamma*error*y

def plot_outputs(y_history, ym_history):
    plt.figure
    plt.plot(tval, y_history)
    plt.plot(tval, ym_history)
    plt.title('True Model and Reference Model')
    plt.legend('y')
    plt.legend('ym')
    plt.xlabel('time')
    plt.ylabel('Outputs')
    plt.grid(True)
    plt.show()

def plot_gains(kr_history, ky_history, kr_star_hist, ky_star_hist):
    plt.figure
    plt.plot(tval, kr_history,'b', 'LineWidth',1)
    plt.plot(tval, kr_star_hist,'b--', 'LineWidth',2)
    plt.plot(tval, ky_history,'r', 'LineWidth',1)
    plt.plot(tval, ky_star_hist,'r--', 'LineWidth',2)
    plt.title('Gains Time history')
    plt.xlabel('time')
    plt.ylabel('Gains')
    plt.legend('kr')
    plt.legend('kr*')
    plt.legend('ky')
    plt.legend('ky*')
    plt.grid(True)
    plt.show()

for t in tval[1:]:
    r = ref(t)
    y = y + np.dot(y_dot(y, r, ky, kr), dt)
    y_history.append(y)

    ym = ym + np.dot(ym_dot(ym, r), dt)
    ym_history.append(ym)
    error = y - ym
    error_history.append(error)
    kr = kr + np.dot(kr_dot(gamma, error, r), dt)
    kr_history.append(kr)
    ky = ky + np.dot(ky_dot(gamma, error, y), dt)
    ky_history.append(ky)

    delta_r = kr - kr_star
    delta_r_hist.append(delta_r)
    delta_y = ky - ky_star
    delta_y_hist.append(delta_y)
```

```
plot_outputs(y_history, ym_history)
plot_gains(kr_history, ky_history, kr_star_hist, ky_star_hist)
```

Problem 2: Shortest path through a grid

a) Dynamic Programming

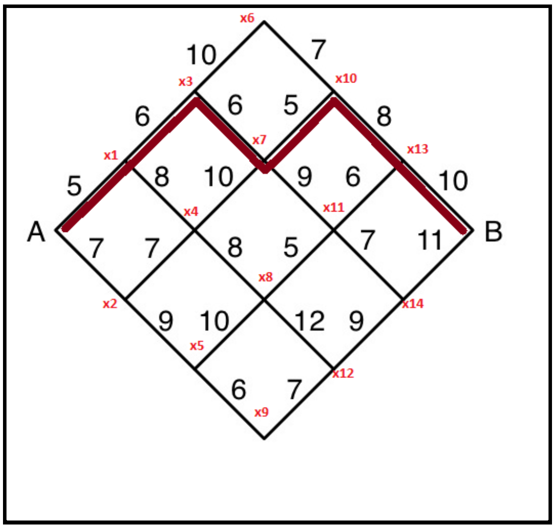

Figure 5

$$J^*(B) = 0,$$

Computation 1,

$$J^*(x_{13}) = 10 + J^*(B) = 10$$

Computation 2,

$$J^*(x_{14}) = 11 + J^*(B) = 11$$

Computation 3 and 4,

$$J^*(x_{11}) = \min\{6 + J^*(x_{13}), 7 + J^*(x_{14})\} = 16$$

Computation 5,

$$J^*(x_{10}) = 8 + J^*(x_{13}) = 18$$

Computation 6 and 7,

$$J^*(x_7) = \min\{5 + J^*(x_{10}), 9 + J^*(x_{11})\} = 23$$

Computation 8,

$$J^*(x_6) = 7 + J^*(x_{10}) = 25$$

Computation 9 and 10,

$$J^*(x_3) = \min\{10 + J^*(x_6), 6 + J^*(x_7)\} = 29$$

Computation 11,

$$J^*(x_4) = 10 + J^*(x_7) = 33$$

Computation 12 and 13,

$$J^*(x_1) = \min\{6 + J^*(x_3), 8 + J^*(x_4)\} = 35$$

Computation 14 and 15,

$$J^*(x_1) = \min\{5 + J^*(x_1), 7 + J^*(x_2)\} = 40$$

Total Cost = 40

A - $x_1 - x_3 - x_7 - x_{10} - x_{13}$- B.

b) For the case of exhaustive search, we have *2n* and the order doesn't matter. Each element from the total *n* can be chosen.

$$C_n^{2n} = \binom{2n}{n} = \frac{(2n)!}{n!\,n!},$$

For *n=3*,

$$C_3^6 = \binom{6}{3} = \frac{6!}{3!\,3!} = \frac{120}{6} = 20.$$

For the DP algorithm the relation is between the number of nodes. To compute the number of nodes by the number of segments (*n*) – nodes = $(n+1)^2$. The number of computations in DP is equal to – *(nodes -1)* or $(n+1)^2 - 1, O(n^2)$.

Problem 3: Machine maintenance

In this problem we have two machine states – Running (R) ou Broke Down (B) – and for each state we have two actios – Running [Maintenance (m), Do Nothing (DN)] or Broke Down [Repair (rp), Replace (r)]. The costs and probabilities:

Maintenance - \$20 – fail:04

Do Nothing - \$0 – fail:0.7

Repair – \$40 – fail:0.4

Replace - \$150 – fail:0

Above we have the Table with set of possibilities for 4 weeks long, considering the new machine (Replaced) at the beginning of first week:

	W1	W2	W3	W4
1	R	R	R	R
2	R	R	R	B
3	R	R	B	R
4	R	R	B	B
5	R	B	R	R
6	R	B	R	B
7	R	B	B	R
8	R	B	B	B

$$Q_k^{\pi}(x,a) = c_k(x,a) + \sum_{x' \in X} p(x'|x,a) J_{k+1}^{\pi}(x')$$

1 – $Q_1^{\pi}(x,m) = -50 + \big(0.4(0-20) + 0.6(100-20)\big) = -10\ (\pi^*)$

$Q_1^{\pi}(x,DN) = -50 + \big(0.7(0) + 0.3(100-0)\big)$ =-20

$Q_2^{\pi}(x,m) = -10 + \big(0.4(0-20) + 0.6(100-20)\big)$ =30 (π^*)

$Q_2^{\pi}(x,DN) = -10 + \big(0.7(0) + 0.3(100-0)\big)$ =20

$Q_3^{\pi}(x,m) = 30 + \big(0.4(0-20) + 0.6(100-20)\big)$ =70 (π^*)

$Q_2^{\pi}(x,DN) = 30 + \big(0.7(0) + 0.3(100-0)\big)$ =60

- $\pi^*[r,m,m,m]$

---

2 – Same as case 1, change in W04,

$Q_3^{\pi}(x,rp) = 30 + \big(0.4(0-40) + 0.6(100-40)\big)$ =50 (π^*)

$Q_3^{\pi}(x,r) = 30 + 100 - 150$=-20

- $\pi^*[r,m,m,rp]$

---

3 – As case 2 change in W03,

$Q_2^{\pi}(x,rp) = -10 + \big(0.4(0-40) + 0.6(100-40)\big)$ =10 (π^*)

$Q_2^{\pi}(x, r) = -10 + 100 - 150$= -60

$Q_3^{\pi}(x, m) = 10 + \big(0.4(0 - 20) + 0.6(100 - 20)\big)$ =50 (π^*)

$Q_3^{\pi}(x, DN) = 10 + \big(0.7(0) + 0.3(100 - 0)\big)$ =40

- $\pi^*[r, m, rp, m]$

---

4 – Like in Case 3, change in W04,

$Q_3^{\pi}(x, m) = 10 + \big(0.4(0 - 40) + 0.6(100 - 40)\big)$ =30 (π^*)

$Q_3^{\pi}(x, r) = 10 + 100 - 150$=-40

- $\pi^*[r, m, rp, rp]$

---

5 - $Q_1^{\pi}(x, rp) = -50 + \big(0.4(0 - 40) + 0.6(100 - 40)\big)$=-30 ($\pi^*$)

$Q_1^{\pi}(x, r) = -50 + 100 - 150$=-100

$Q_2^{\pi}(x, m) = -30 + \big(0.4(0 - 20) + 0.6(100 - 20)\big)$ =10 (π^*)

$Q_2^{\pi}(x, DN) = -30 + \big(0.7(0) + 0.3(100 - 0)\big)$ =0

$Q_3^{\pi}(x, m) = 10 + \big(0.4(0 - 20) + 0.6(100 - 20)\big)$ =50 (π^*)

$Q_2^{\pi}(x, DN) = 10 + \big(0.7(0) + 0.3(100 - 0)\big)$ =40

- $\pi^*[r, rp, m, m]$

6- $Q_1^{\pi}(x, rp) = -50 + \big(0.4(0 - 40) + 0.6(100 - 40)\big)$=-30 ($\pi^*$)

$Q_1^{\pi}(x, r) = -50 + 100 - 150$= -100

$Q_2^{\pi}(x, m) = -30 + \big(0.4(0 - 20) + 0.6(100 - 20)\big)$ =10 (π^*)

$Q_2^{\pi}(x, DN) = -30 + \big(0.7(0) + 0.3(100 - 0)\big)$ =0

$Q_3^{\pi}(x, rp) = 10 + \big(0.4(0 - 40) + 0.6(100 - 40)\big)$ =30 (π^*)

$Q_3^{\pi}(x, r) = 10 + 100 - 150$=-40

- $\pi^*[r, rp, m, rp]$

---

7 – Like 6, change in W3,

$Q_2^{\pi}(x, rp) = -30 + \big(0.4(0 - 40) + 0.6(100 - 40)\big)$ =-10 (π^*)

$Q_2^{\pi}(x, r) = -30 + 100 - 150$=-80

$Q_3^{\pi}(x, m) = -10 + \big(0.4(0 - 20) + 0.6(100 - 20)\big)$ =30 (π^*)

$Q_3^{\pi}(x, DN) = -10 + \big(0.7(0) + 0.3(100 - 0)\big)$ =20

- $\pi^*[r, rp, rp, m]$

---

8 – Like 7, change only in W4,

$Q_3^{\pi}(x, rp) = -10 + \big(0.4(0-40) + 0.6(100-40)\big)$ =10 (π^*)

$Q_3^{\pi}(x, r) = -10 + 100 - 150$=-60

- $\pi^*[r, m, rp, rp]$

Problem 4: Markovian drone

a) Heatmap for *V(x)*

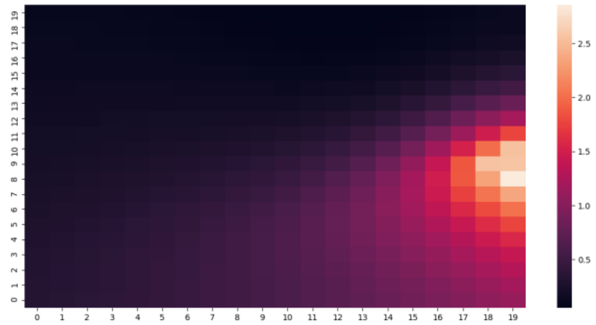

Figure 6 – Cost function

b.1) Policy heatmap

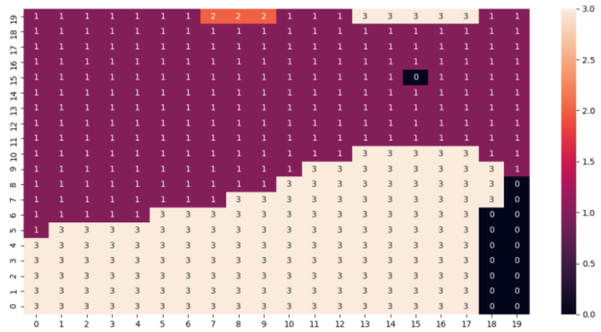

Figure 7 – Policy.

b.2) Drone Trajectory

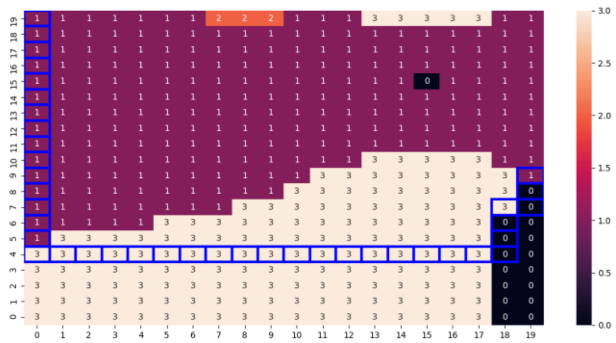

Figure 8 – Trajectory of drone (squares with blue edge).

Policy task

For each state, the permitted actions were evaluated by the cost function, the actions most valuable will be chosen for optimal policy.

Python code: Grid

```
import numpy as np


class Grid:

    # action (0=up, 1=down, 2=left, 3=right)

    def __init__(self, n):
        self.actions = [0, 1, 2, 3]
        self.n = n
        self.all_states = self.grid()
        self.ns = n*n
        self.na = len(self.actions)
        self.rewards = {}
        self.aS = {}

    def grid(self):
        states = []
        for x1 in range(self.n):
            for x2 in range(self.n):
                states.append((x1,x2))
        return states

    def actions_space(self, x_goal):
        for x in self.all_states:
            act = ()
            for a in self.actions:
                x_next = self.execute(a, x)
                if self.validate(x_next):
                    act = act + (a,)
            self.aS[x] = act

    def set_rewards(self, x_goal):
        for x in self.all_states:
            if x == x_goal:
                self.rewards[x] = 1
            else:
                self.rewards[x] = 0

    def prob(self, a, x):
        x_next = self.execute(a, x)
        if not(self.validate(x_next)):
                x_next = x

        return x_next

    def validate(self, x):
```

```
        x1, x2 = x
        if (x1 >= self.n or x1 < 0) or (x2 >= self.n or x2 < 0):
            return False
        return True

    def execute(self, a,x):
        x1, x2 = x
        if a == 0:
            x_next = (x1, x2 + 1)
        elif a == 1:
            x_next = (x1, x2 - 1)
        elif a == 2:
            x_next = (x1 - 1, x2)
        else:# a == 3
            x_next = (x1 + 1, x2)

        return x_next
```

Python code: Value Iteration

```
import numpy as np
import matplotlib.pyplot as plt
from grid import Grid as G
import seaborn as sb
import pandas as pd
import math as m
from matplotlib.patches import Rectangle

n=20
env = G(n)

x_eye=(15,15)
x_goal=(19,9)

env.actions_space(x_goal)
env.set_rewards(x_goal)

V = {}
for x in env.all_states:
    V[x] = 0
V[x_goal] = 0

policy = {}
for x in env.aS.keys():
    policy[x] = np.random.choice(env.aS[x])

def value_iteration(env, epsilon=0.0001, discount=0.95, sigma=10):
```

```python
    def next_step(V, x):
        x1, x2 = x
        x1_eye, x2_eye = x_eye
        w = np.exp(-((x1 - x1_eye)**2 + (x2 - x2_eye)**2)/(2*(sigma**2)))

        v = np.zeros(4)
        for a in env.aS[x]:
            x_next = env.prob(a, x)
            for r_act in env.aS[x]:
                if (r_act == a):
                    p = (1 - w + w/len(env.aS[x]))
                    r = env.rewards[x_next]
                    v_next = V[x_next]
                else:
                    x_next_rand = env.prob(r_act, x)
                    p = w/len(env.aS[x])
                    r = env.rewards[x_next_rand]
                    v_next = V[x_next_rand]

                v[a] += p*(r + discount*v_next)

        return v

    iteration = 0
    while True :
        delta = 0
        for x in env.all_states:
            Q = next_step(V, x)
            best_v_action = max(Q)
            delta = max(delta, np.abs(V[x] - best_v_action))
            V[x] = best_v_action
            policy[x] = np.argmax(Q)

        if delta < epsilon:
            break
        iteration += 1

    print(iteration)

    return V, policy

def plot_heatmap(data, ant):
    m = np.array([data[key] for key in data.keys()]).reshape((n, n)).T
    ax = sb.heatmap(np.round(m, 3), annot=ant)
    ax.invert_yaxis()
    p = path(data, (0,19))
    for t in p:
```

```
        ax.add_patch(Rectangle(t, 1, 1, fill=False, edgecolor='blue', lw=
3))
    plt.show()

def path(policy, x_start):
    p = []
    x = x_start
    N = 100
    p.append(x_start)
    iteration = 0
    while iteration <= N:
        if x == x_goal:
            break
        x = env.prob(policy[x], x)
        p.append(x)
        iteration += 1
    return p

V, policy = value_iteration(env)
plot_heatmap(V, False)
plot_heatmap(policy, True)
```

Problem 5: cart-Pole balance

a) Given $s^* = (0, \pi, 0,0), u^* = 0, \Delta s_k = s_k - s^*, s_{k+1} \approx s_k + \Delta t f(s_k, u_k)$, linearize about (s^*, u^*) and yields LTI system, $\Delta s_{k+1} \approx A\Delta s_k + Bu_k$.

$$s_{k+1} \approx \left(s + \Delta t f(s,u)\right)|_{(s^*,u^*)} + \left(I_4 + \Delta t \frac{\partial f(s,u)}{\partial s}\right)|_{(s^*,u^*)}(s - s^*)$$
$$+ \left(\frac{\partial s}{\partial u} + \Delta t \frac{\partial f(s,u)}{\partial u}\right)|_{(s^*,u^*)}(u - u^*).$$

Where $\Delta t f(s^*, u^*) = 0, \frac{\partial s^*}{\partial u} = 0, u^*$:

$$s_{k+1} - s^* \approx \left(I_4 + \Delta t \frac{\partial f(s^*,u^*)}{\partial s}\right)\Delta s_k + \Delta t \frac{\partial f(s^*,u^*)}{\partial u} u_k,$$

$$s_{k+1} - s^* = \Delta s_{k+1},$$

$$\mathrm{A} = I_4 + \Delta t \frac{\partial f(s^*,u^*)}{\partial s} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{m_p g}{m_c} & 1 & 0 \\ 0 & \frac{(m_c+m_p)g}{m_c l} & 0 & 1 \end{bmatrix},$$

$$B = \Delta t \frac{\partial f(s^*,u^*)}{\partial u} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{m_c l} \end{bmatrix},$$

Finally,

$$\Delta s_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \frac{m_p g}{m_c} & 1 & 0 \\ 0 & \frac{(m_c+m_p)g}{m_c l} & 0 & 1 \end{bmatrix} \Delta s_k + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{m_c l} \end{bmatrix} u_k.$$

b) LQR controller

k_∞ results,

[[0. 0. 0. 0.]]

[[0. -0.01 -0.01 -0.01]]

[[-0. -0.04 -0.02 -0.02]]

[[-0. -0.09 -0.03 -0.04]]

[[-0.01 -0.18 -0.04 -0.06]]

[[-0.01 -0.33 -0.06 -0.11]]

[[-0.02 -0.59 -0.07 -0.18]]

```
[[-0.02 -1.05 -0.09 -0.32]]
[[-0.03 -1.86 -0.11 -0.55]]
[[-0.04 -3.3  -0.13 -0.97]]
[[-0.05 -5.82 -0.16 -1.7 ]]
[[ -0.06 -10.18  -0.19  -2.97]]
[[ -0.07 -17.54  -0.23  -5.11]]
[[ -0.08 -29.44  -0.26  -8.57]]
[[ -0.09 -47.39  -0.28 -13.79]]
[[ -0.09 -71.82  -0.29 -20.91]]
[[-8.0000e-02 -1.0082e+02 -2.7000e-01 -2.9360e+01]]
[[-6.0000e-02 -1.3014e+02 -2.2000e-01 -3.7910e+01]]
[[-4.0000e-02 -1.5536e+02 -1.5000e-01 -4.5270e+01]]
[[-1.0000e-02 -1.7425e+02 -7.0000e-02 -5.0780e+01]]
[[ 1.000e-02 -1.870e+02 -0.000e+00 -5.451e+01]]
[[ 3.0000e-02 -1.9504e+02  6.0000e-02 -5.6860e+01]]
[[ 4.0000e-02 -1.9994e+02  1.2000e-01 -5.8300e+01]]
[[ 6.0000e-02 -2.0288e+02  1.7000e-01 -5.9160e+01]]
[[ 7.0000e-02 -2.0467e+02  2.2000e-01 -5.9690e+01]]
[[ 9.0000e-02 -2.0579e+02  2.6000e-01 -6.0020e+01]]
[[ 1.0000e-01 -2.0653e+02  3.0000e-01 -6.0240e+01]]
[[ 1.1000e-01 -2.0706e+02  3.5000e-01 -6.0410e+01]]
[[ 1.3000e-01 -2.0748e+02  3.9000e-01 -6.0540e+01]]
[[ 1.4000e-01 -2.0784e+02  4.4000e-01 -6.0650e+01]]
[[ 1.5000e-01 -2.0818e+02  4.9000e-01 -6.0750e+01]]
[[ 1.700e-01 -2.085e+02  5.300e-01 -6.086e+01]]
[[ 1.8000e-01 -2.0883e+02  5.9000e-01 -6.0960e+01]]
[[ 2.0000e-01 -2.0917e+02  6.4000e-01 -6.1070e+01]]
[[   0.21 -209.52    0.7   -61.18]]
[[   0.23 -209.88    0.76  -61.29]]
[[   0.24 -210.25    0.82  -61.41]]
[[   0.26 -210.64    0.88  -61.53]]
```

[[0.27 -211.05 0.95 -61.66]]

[[0.29 -211.47 1.02 -61.79]]

[[0.31 -211.9 1.09 -61.93]]

[[0.32 -212.35 1.16 -62.07]]

[[0.34 -212.82 1.23 -62.22]]

[[0.35 -213.29 1.31 -62.37]]

[[0.37 -213.78 1.39 -62.53]]

[[0.39 -214.28 1.47 -62.68]]

[[0.4 -214.79 1.55 -62.84]]

[[0.42 -215.3 1.64 -63.01]]

[[0.44 -215.82 1.72 -63.17]]

[[0.45 -216.35 1.8 -63.34]]

[[0.47 -216.89 1.89 -63.51]]

[[0.48 -217.42 1.98 -63.68]]

[[0.5 -217.96 2.06 -63.85]]

[[0.51 -218.5 2.15 -64.02]]

[[0.53 -219.03 2.23 -64.19]]

[[0.54 -219.56 2.32 -64.36]]

[[0.56 -220.09 2.4 -64.52]]

[[0.57 -220.61 2.48 -64.69]]

[[0.58 -221.13 2.56 -64.85]]

[[0.59 -221.63 2.64 -65.01]]

[[0.6 -222.13 2.72 -65.17]]

[[0.61 -222.61 2.8 -65.32]]

[[0.62 -223.09 2.87 -65.47]]

[[0.63 -223.55 2.94 -65.62]]

[[0.64 -224. 3.01 -65.76]]

[[0.65 -224.43 3.08 -65.9]]

[[0.66 -224.85 3.15 -66.03]]

[[0.67 -225.25 3.21 -66.16]]

[[0.67 -225.64 3.27 -66.28]]

[[0.68 -226.02   3.33 -66.4]]

[[0.68 -226.37   3.38 -66.51]]

[[0.69 -226.71   3.44 -66.62]]

[[0.69 -227.04   3.49 -66.72]]

[[0.7 -227.35   3.54 -66.82]]

[[0.7 -227.65   3.58 -66.92]]

[[0.71 -227.92   3.62 -67.]]

[[0.71 -228.19   3.66 -67.09]]

[[0.71 -228.44   3.7   -67.17]]

[[0.71 -228.67   3.74 -67.24]]

[[0.72 -228.89   3.77 -67.31]]

[[0.72 -229.1   3.8   -67.38]]

[[0.72 -229.29   3.83 -67.44]]

[[0.72 -229.47   3.86 -67.49]]

[[0.72 -229.64   3.89 -67.55]]

[[0.72 -229.8   3.91 -67.6]]

[[0.72 -229.95   3.93 -67.64]]

[[0.72 -230.08   3.95 -67.69]]

[[0.72 -230.21   3.97 -67.73]]

[[0.72 -230.32   3.99 -67.76]]

[[0.72 -230.43   4.   -67.8]]

[[0.72 -230.53   4.02 -67.83]]

[[0.72 -230.62   4.03 -67.86]]

[[0.72 -230.7   4.04 -67.88]]

[[0.72 -230.78   4.06 -67.91]]

[[0.72 -230.85   4.07 -67.93]]

[[0.72 -230.91   4.08 -67.95]]

[[0.72 -230.97   4.08 -67.97]]

[[0.72 -231.02   4.09 -67.98]]

[[0.72 -231.07   4.1   -68.]]

[[0.72 -231.11   4.1   -68.01]]

[[0.72 -231.15 4.11 -68.02]]

[[0.72 -231.18 4.11 -68.04]]

[[0.72 -231.21 4.12 -68.04]]

[[0.71 -231.24 4.12 -68.05]]

[[0.71 -231.27 4.13 -68.06]]

[[0.71 -231.29 4.13 -68.07]]

[[0.71 -231.31 4.13 -68.07]]

[[0.71 -231.33 4.13 -68.08]]

[[0.71 -231.34 4.14 -68.08]]

[[0.71 -231.35 4.14 -68.09]]

[[0.71 -231.37 4.14 -68.09]]

[[0.71 -231.38 4.14 -68.1]]

[[0.71 -231.38 4.14 -68.1]]

[[0.71 -231.39 4.14 -68.1]]

[[0.71 -231.4 4.14 -68.1]]

[[0.71 -231.41 4.15 -68.11]]

[[0.71 -231.41 4.15 -68.11]]

[[0.71 -231.42 4.15 -68.11]]

[[0.71 -231.42 4.15 -68.11]]

[[0.71 -231.42 4.15 -68.11]]

[[0.71 -231.43 4.15 -68.11]]

[[0.71 -231.43 4.15 -68.11]]

[[0.71 -231.43 4.15 -68.11]]

[[0.71 -231.43 4.15 -68.11]]

[[0.71 -231.44 4.15 -68.12]]

[[0.71 -231.44 4.15 -68.12]]

[[0.71 -231.44 4.15 -68.12]]

[[0.71 -231.44 4.15 -68.12]]

[[0.71 -231.45 4.15 -68.12]]

[[0.71 -231.45 4.15 -68.12]]

[[0.71 -231.45 4.15 -68.12]]

[[0.71 -231.46 4.15 -68.12]]

[[0.71 -231.46 4.15 -68.12]]

[[0.71 -231.46 4.15 -68.12]]

[[0.71 -231.47 4.15 -68.12]]

[[0.71 -231.47 4.16 -68.13]]

[[0.71 -231.47 4.16 -68.13]]

[[0.71 -231.48 4.16 -68.13]]

[[0.71 -231.48 4.16 -68.13]]

[[0.71 -231.49 4.16 -68.13]]

[[0.71 -231.49 4.16 -68.13]]

[[0.71 -231.5 4.16 -68.13]]

[[0.71 -231.5 4.16 -68.14]]

[[0.71 -231.51 4.16 -68.14]]

[[0.71 -231.51 4.16 -68.14]]

[[0.71 -231.52 4.16 -68.14]]

[[0.71 -231.52 4.16 -68.14]]

[[0.71 -231.53 4.17 -68.14]]

[[0.72 -231.53 4.17 -68.15]]

[[0.72 -231.54 4.17 -68.15]]

[[0.72 -231.55 4.17 -68.15]]

[[0.72 -231.55 4.17 -68.15]]

[[0.72 -231.56 4.17 -68.15]]

[[0.72 -231.56 4.17 -68.16]]

[[0.72 -231.57 4.17 -68.16]]

[[0.72 -231.58 4.17 -68.16]]

[[0.72 -231.58 4.18 -68.16]]

[[0.72 -231.59 4.18 -68.16]]

[[0.72 -231.6 4.18 -68.17]]

[[0.72 -231.6 4.18 -68.17]]

[[0.72 -231.61 4.18 -68.17]]

[[0.72 -231.62 4.18 -68.17]]

[[0.72 -231.62   4.18 -68.17]]

[[0.72 -231.63   4.18 -68.18]]

[[0.72 -231.64   4.18 -68.18]]

[[0.72 -231.64   4.19 -68.18]]

[[0.72 -231.65   4.19 -68.18]]

[[0.72 -231.66   4.19 -68.18]]

[[0.72 -231.66   4.19 -68.19]]

[[0.72 -231.67   4.19 -68.19]]

[[0.72 -231.67   4.19 -68.19]]

[[0.72 -231.68   4.19 -68.19]]

[[0.72 -231.69   4.19 -68.19]]

[[0.72 -231.69   4.19 -68.2]]

[[0.72 -231.7   4.19 -68.2]]

[[0.72 -231.7   4.2   -68.2]]

[[0.72 -231.71   4.2   -68.2]]

[[0.72 -231.71   4.2   -68.2]]

[[0.72 -231.72   4.2   -68.2]]

[[0.73 -231.72   4.2   -68.21]]

[[0.73 -231.73   4.2   -68.21]]

[[0.73 -231.73   4.2   -68.21]]

[[0.73 -231.74   4.2   -68.21]]

[[0.73 -231.74   4.2   -68.21]]

[[0.73 -231.75   4.2   -68.21]]

[[0.73 -231.75   4.2   -68.22]]

[[0.73 -231.76   4.2   -68.22]]

[[0.73 -231.76   4.2   -68.22]]

[[0.73 -231.77   4.21 -68.22]]

[[0.73 -231.77   4.21 -68.22]]

[[0.73 -231.77   4.21 -68.22]]

[[0.73 -231.78   4.21 -68.22]]

[[0.73 -231.78   4.21 -68.22]]

[[0.73 -231.78   4.21 -68.23]]

[[0.73 -231.79   4.21 -68.23]]

[[0.73 -231.79   4.21 -68.23]]

[[0.73 -231.79   4.21 -68.23]]

[[0.73 -231.8   4.21 -68.23]]

[[0.73 -231.8   4.21 -68.23]]

[[0.73 -231.8   4.21 -68.23]]

[[0.73 -231.81   4.21 -68.23]]

[[0.73 -231.81   4.21 -68.23]]

[[0.73 -231.81   4.21 -68.23]]

[[0.73 -231.81   4.21 -68.23]]

[[0.73 -231.82   4.21 -68.24]]

[[0.73 -231.82   4.21 -68.24]]

[[0.73 -231.82   4.21 -68.24]]

[[0.73 -231.82   4.21 -68.24]]

[[0.73 -231.82   4.21 -68.24]]

[[0.73 -231.82   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.83   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

[[0.73 -231.84   4.22 -68.24]]

```
[[   0.73 -231.84    4.22  -68.24]]
[[   0.73 -231.84    4.22  -68.24]]
[[   0.73 -231.84    4.22  -68.24]]
[[   0.73 -231.85    4.22  -68.24]]
[[   0.73 -231.85    4.22  -68.24]]
[[   0.73 -231.85    4.22  -68.24]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
```

```
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
```

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

[[0.73 -231.85   4.22 -68.25]]

```
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
```

```
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
```

```
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
[[   0.73 -231.85    4.22  -68.25]]
```

[[0.73 -231.85 4.22 -68.25]]

[[0.73 -231.85 4.22 -68.25]]

[[0.73 -231.85 4.22 -68.25]]

[[0.73 -231.85 4.22 -68.25]]

[[0.73 -231.85 4.22 -68.25]]

Python Code:

```python
import numpy as np

def riccati():
    dt = 0.1
    mp = 2.0
    mc = 10.0
    l = 1.0
    g = 9.81
    Pk = np.zeros((4,4))
    Kk = np.zeros((1,4))
    Q = np.eye(4,4)
    R = 1
    A = np.matrix([[1,0,dt,0],[0,1,0,dt],[0,dt*mp*g/mc,1,0],[0,dt*(mc+mp)
*g/(mc*l),0,1]])
    B = np.array([0,0,dt/mc,dt/(mc*l)]).reshape(4,1)
    while True:
        Pk_adv = Pk
        Kk = (np.linalg.inv(-(R + (B.T)@Pk_adv@B))*(B.T)@Pk_adv@A)
        Pk = (Q + A.transpose()@Pk_adv@(A + B@Kk))
        print(Kk.round(2))
        if np.linalg.norm(Pk_adv - Pk) < 10**(-4):
            break

    return Kk

K = riccati()
```

c) Simulate the system,

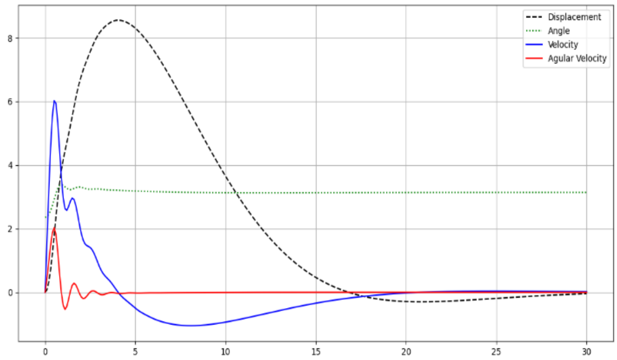

Figure 9 - $x, \theta, \dot{x}, \dot{\theta}$.

d) Noise added, $\mu = 0$ and $cov = diag(0, 0,\ 10^{-4}, 10^{-4}), t \in [0,30]$.

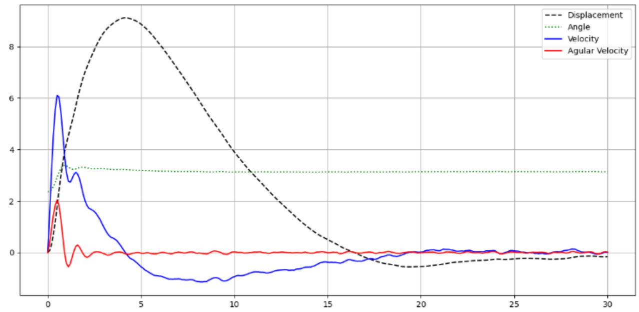

Figure 10 - $x, \theta, \dot{x}, \dot{\theta}$ noise added.

Python Code:

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt

import animations as an

def cartpole(s,t,u):
    mp = 2.0
    mc = 10.0
    l = 1.0
    g = 9.81
    _, teta, x_dot, teta_dot = s
    x_ddot = (mp*np.sin(teta)*(l*teta_dot**2 + g*np.cos(teta)) + u)/(mc +
 mp*(np.sin(teta))**2)
    teta_ddot = -
((mc + mp)*g*np.sin(teta) + mp*l*(teta_dot**2)*np.sin(teta)*np.cos(teta)
+ u*np.cos(teta))/((mc + mp*(np.sin(teta))**2)*l)
    dsdt = [x_dot, teta_dot, x_ddot, teta_ddot]
```

```
    return dsdt

def noise(mean, cov):
    cov_matrix = np.diag(cov)
    w = np.random.multivariate_normal(mean, cov_matrix, 1)
    return w

def simulate(s0, animated, add_noise):
    tf = 30.0
    dt = 0.1
    t = np.linspace(0, tf, int(tf/dt) + 1)
    kinf = [0.7291397,  -231.85419281,    4.21967188,  -68.24742825]
    w =  noise(np.array([0,0,0,0]), np.array([0, 0 ,10**(-4),10**(-4)]))
    s_star = np.array([0, np.pi, 0, 0])
    if add_noise:
        s = [s0 + w[0]]
    else:
        s = [s0]
    u = [kinf @ (s[0] - s_star)]

    for k in range(len(t)-1):
        if add_noise:
            w =  noise(np.array([0,0,0,0]), np.array([0, 0 ,10**(-
4),10**(-4)]))
            s.append((odeint(cartpole, s[k], t[k:k+2],(u[k],))[1] + w[0])
)
            u.append(kinf @ (s[k] - s_star))
        else:
            s.append(odeint(cartpole, s[k], t[k:k+2],(u[k],))[1])
            u.append(kinf @ (s[k] - s_star))

    _, ax = plt.subplots()
    ax.plot(t, np.asanyarray(s)[:,0], 'k--', label='Displacement')
    ax.plot(t, np.asanyarray(s)[:,1], 'g:', label='Angle')
    ax.plot(t, np.asanyarray(s)[:,2], 'b', label='Velocity')
    ax.plot(t, np.asanyarray(s)[:,3], 'r', label='Agular Velocity')
    plt.grid(True)
    ax.legend()
    if animated:
        _, _ = an.animate_cartpole(t, np.asanyarray(s)[:,0], np.asanyarra
y(s)[:,1])

    plt.show()

simulate(np.array([0, 3*np.pi/4, 0, 0]), True, False)
```