

# **Advanced Control for Robotics: Homework #1**

Shang Yangxing

January 17, 2021

# 1 ODE and Its Simulation

## 1.1 Equation of Pendulum Motions

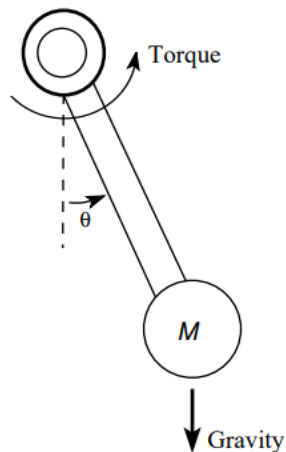


Figure 1: pendulum model

By applying the Newton's law of dynamics, a pendulum with no external force can be formulated as:

$$ml^2\ddot{\theta} + ml^2\alpha\dot{\theta} + mgl\sin\theta - T = 0. \quad (1)$$

in which,

$m$  is mass of the ball

$l$  is length of the rod

$\alpha$  is the damping constant

$g$  is the gravitational constant

$\theta$  is angle measured between the rod and the vertical axis

$T$  is torque of the joint, which is also the control input  $u$

to a system of two first order equation by letting  $x_1 = \theta$ ,  $x_2 = \dot{\theta}$ :

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = -\frac{g}{l}\sin x_1 - \alpha x_2 + \frac{T}{ml^2}. \quad (2)$$

Written in standard state-space form:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin x_1 - \alpha x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} T \quad (3)$$

$$\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x} \quad (4)$$

## 1.2 Simulation of Pendulum

When assuming  $m = l = 1$  with proper unit, equation (3) can be simplified as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -g \sin x_1 - \alpha x_2 + T \end{bmatrix} \quad (5)$$

according the equation, we code the simulation as following:

```
1 import numpy as np
2 from scipy.integrate import odeint
3 import matplotlib.pyplot as plt
4
5 def pendulum(x, t, g, alpha, T):
6     '''
7     pendulum system vector-space function
8     '''
9     x1, x2 = x
10    dxdt = [x2, -g*np.sin(x1) - alpha*x2 + T ]
11    return dxdt
12
13 ## initial condition
14 g = 9.8 # gravitational constant
15
16 # damping constant alpha collection of two different cases
17 alpha_c = [0.3, 0.7]
18 T = 0 # the control input
19
20 # initial theta collection of two different cases
21 x1_0_c = [np.pi*3/4, np.pi/4]
22 x2_0 = 0 # initial omega
23
24 ## simulation setup
25 t = np.linspace(0, 9.9, 400)
26 # y = [] # the output collection of four cases
27
28 plt.subplots(2, 2, sharex='all', sharey='all', figsize=(14, 8))
29 # plt.figure()
30
31 for i in range(4):
32     '''
33     four cases
34     choose x1_0 with rem:
35         when i = 0 or 2, x1_0 is in the first case;
36         when i = 1 or 3, in another one
37     choose alpha with mod:
38         when i = 0 or 1, alpha is in the first case;
39         when i = 2 or 3, in another one
40     '''
41    x0 = [x1_0_c[i%2], x2_0]
42    alpha = alpha_c[i//2]
```

```

43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

```

## solve
y = odeint(pendulum, x0, t, args=(g, alpha, T))

## plot
plt.subplot(2, 2, i+1)
plt.plot(t, y[:, 0], label='x1:theta')
plt.plot(t, y[:, 1], label='x2:omega')
plt.title('x1_0={:.2f},x2_0={:.2f},alpha={:.2f},T={:.2f}'\
          .format(x0[0], x0[1], alpha, T))
plt.legend(loc='best')
plt.ylim(-6, 6)
if(i>=2):plt.xlabel('time')
plt.grid()

## save and show
plt.savefig(r'./HW1/img/pendulumSim.png')
plt.show()

```

and getting the output as:

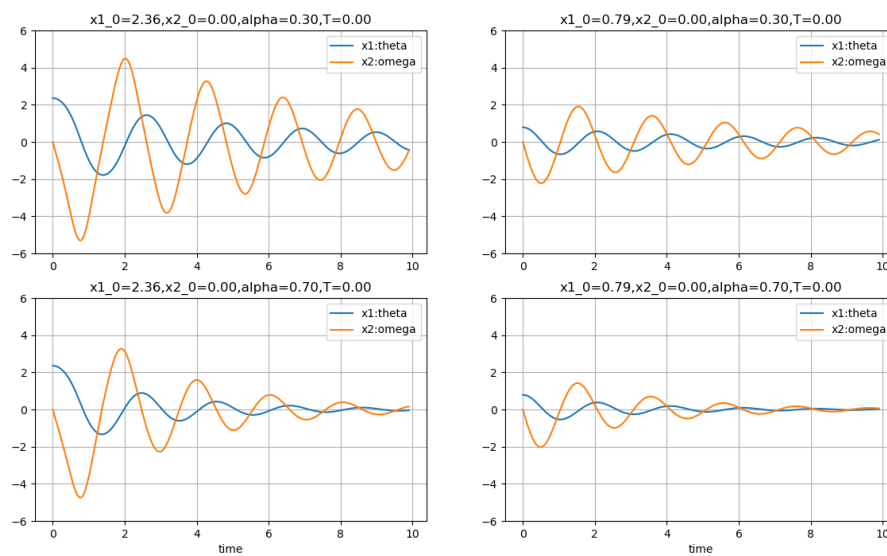


Figure 2: pendulum simulation output

- 2 Matrix calculus
- 3 Inner product
- 4 Some linear algebra
- 5 Gradient Flow