Advanced Control for Robotics: Homework #1

Shang Yangxing

January 18, 2021

1 ODE and Its Simulation

1.1 Equation of Pendulum Motions

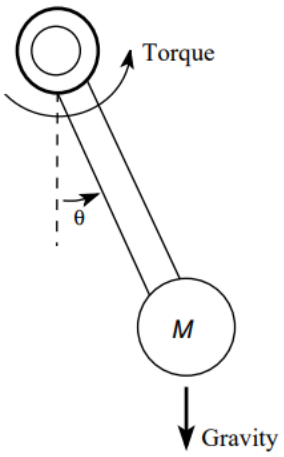

Figure 1: pendulum model

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

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

in which,
m is mass of the ball
l is length of the rod
α is the damping constant
g is the gravitational constant
θ 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 $x_1 = \theta$, $x_2 = \dot{\theta}$:

$$\dot{x_1} = x_2, \quad \dot{x_2} = -\frac{g}{l}\sin x_1 - \alpha x_2 + \frac{T}{ml^2}. \tag{2}$$

Written in standard state-space form:

$$\dot{\boldsymbol{x}} = \begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin x_1 - \alpha x_2\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix} T \tag{3}$$

$$\boldsymbol{y} = \begin{bmatrix}x_1\\ x_2\end{bmatrix} = \boldsymbol{x} \tag{4}$$

1.2 Simulation of Pendulum

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

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2 \\ -g \sin x_1 - \alpha x_2 + T \end{bmatrix} \tag{5}$$

according the equation, we code the simulation as following:

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

def pendulum(x, t, g, alpha, T):
    '''
    pendulum system vector-space function
    '''
    x1, x2 = x
    dxdt = [x2, -g*np.sin(x1) - alpha*x2 + T ]
    return dxdt

##  inital condition
g = 9.8  # gravitational constant

# damping constant alpha collection of two different cases
alpha_c = [0.3, 0.7]
T = 0  # the control input

# inital theta collection of two different cases
x1_0_c = [np.pi*3/4, np.pi/4]
x2_0 = 0  # inital omega

## simulation setup
t = np.linspace(0, 9.9, 400)
# y = []  # the output collection of four cases

plt.subplots(2, 2, sharex='all', sharey='all', figsize=(14, 8))
# plt.figure()

for i in range(4):
    '''
    four cases
    choose x1_0 with rem:
        when i = 0 or 2, x1_0 is in the first case;
        when i = 1 or 3, in another one
    choose alpha with mod:
        when i = 0 or 1, alpha is in the first case;
        when i = 2 or 3, in another one
    '''
    x0 = [x1_0_c[i%2], x2_0]
    alpha = alpha_c[i//2]
```

```
43
44      ## solve
45      y = odeint(pendulum, x0, t, args=(g, alpha, T))
46
47      ## plot
48      plt.subplot(2, 2, i+1)
49      plt.plot(t, y[:, 0], label='x1:theta')
50      plt.plot(t, y[:, 1], label='x2:omega')
51      plt.title('x1_0={:.2f},x2_0={:.2f},alpha={:.2f},T={:.2f}'\
52              .format(x0[0], x0[1], alpha, T))
53      plt.legend(loc='best')
54      plt.ylim(-6, 6)
55      if(i>=2):plt.xlabel('time')
56      plt.grid()
57
58  ## save and show
59  plt.savefig(r'./HW1/img/pendulumSim.png')
60  plt.show()
```

and getting the output as:

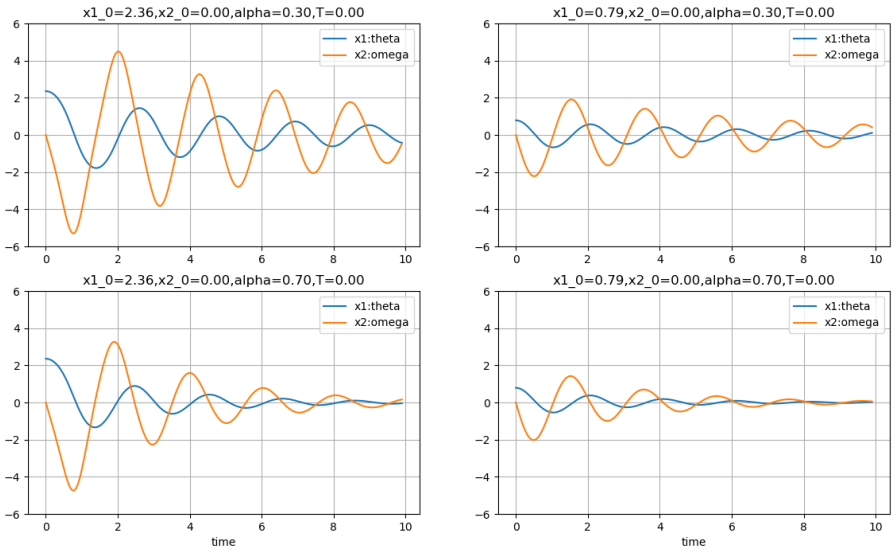

Figure 2: pendulum simulation output

2 Matrix calculus

2.1 Tutorial

$$\frac{\partial}{\partial X}f(X) = \begin{bmatrix} \frac{\partial f(X)}{\partial X_{11}} & \cdots & \frac{\partial f(X)}{\partial X_{1m}} \\ \vdots & \frac{\partial f(X)}{\partial X_{ij}} & \vdots \\ \frac{\partial f(X)}{\partial X_{n1}} & \cdots & \frac{\partial f(X)}{\partial X_{nm}} \end{bmatrix} \tag{6}$$

Derivative of scalar function $f(X)$ can be calculated by taking derivatives of the scalar function with respect to each entry X_{ij} of the matrix X separately, showing as above equation (6).

Scalar function $f(X)$ project matrix variable $X \in \mathbb{R}^{n\times m}$ to a scalar $y \in \mathbb{R}^1$, so its derivative is the partial derivative, except that its results are arranged in form of a matrix, who has the same shape as X.

For instance, let's say $X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$, $f(X) = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$. So $y = f(X) = X_{11} + X_{12} + X_{21} + X_{22}$. And the partial derivative of $f(X)$ is

$$\frac{\partial}{\partial X}f(X) = \begin{bmatrix} \frac{\partial f(X)}{\partial X_{11}} & \frac{\partial f(X)}{\partial X_{12}} \\ \\ \frac{\partial f(X)}{\partial X_{21}} & \frac{\partial f(X)}{\partial X_{22}} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{7}$$

2.2 Derivative of Trace

$$\begin{aligned} \frac{\partial}{\partial X}tr(AX) &= \frac{\partial}{\partial X}tr(\begin{bmatrix} A_{11}X_{11} & \cdots & A_{1m}X_{m1} \\ \vdots & A_{ij}X_{ji} & \vdots \\ A_{n1}X_{1n} & \cdots & A_{nm}X_{mn} \end{bmatrix}) \\ &= \frac{\partial}{\partial X}(A_{11}X_{11} + \cdots + A_{ij}X_{ji} + \cdots + A_{nm}X_{mn}) \\ &= \begin{bmatrix} A_{11} & \cdots & A_{n1} \\ \vdots & A_{ji} & \vdots \\ A_{1m} & \cdots & X_{mn} \end{bmatrix} = A^T \end{aligned} \tag{8}$$

in which, $\frac{\partial}{\partial X_{ij}}(A_{11}X_{11} + \cdots + A_{ij}X_{ji} + \cdots + A_{nm}X_{mn}) = A_{ji}$

2.3 Derivation

According to *The Matrix Cookbook* equation (81), we have

$$\frac{\partial x^T Q x}{\partial x} = (Q + Q^T)x \tag{9}$$

and we can derive that

$$\frac{\partial tr(xx^T)}{\partial x} = \frac{\partial}{\partial x}(x_1^2 + x_2^2 + \cdots + x_n^2) = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_n \end{bmatrix} = 2x \tag{10}$$

comprehensive above, we get

$$\begin{aligned}\frac{\partial}{\partial x} f(x) &= \frac{\partial x^T Q x}{\partial x} + \frac{\partial tr(x x^T)}{\partial x} \\ &= (Q + Q^T)x + 2x\end{aligned} \tag{11}$$

3 Inner product

4 Some linear algebra

5 Gradient Flow