

```
1: import numpy as np
2: import random
3: import pylab
4:
5:
6: def f(x, t):
7:     return -x**3 + np.sin(t)
8:
9:
10: def RungeKutta(func, init, a, b, h=0.1):
11:     # h = abs(b - a) / N
12:     X = []
13:     T = np.arange(a, b, h)
14:     x = init
15:     for t in T:
16:         X.append(x)
17:         k1 = h * func(x, t)
18:         k2 = h * func(x + k1 / 2, t + h / 2)
19:         k3 = h * func(x + k2 / 2, t + h / 2)
20:         k4 = h * func(x + k3, t + h)
21:         x += (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
22:     return T, X
23:
24:
25: def RungeKutta2(f1, f2, x_init, y_init, a, b, h=0.1):
26:     X = []
27:     Y = []
28:     T = np.arange(a, b, h)
29:     x = x_init
30:     y = y_init
31:     for t in T:
32:         # TODO REmove THis!
33:         if (x > np.pi):
34:             x -= 2 * np.pi
35:         if (x < -np.pi):
36:             x += 2 * np.pi
37:         X.append(x)
38:         Y.append(y)
39:         k1 = h * f1(x, y, t)
40:         l1 = h * f2(x, y, t)
41:         k2 = h * f1(x + k1 / 2, y + l1 / 2, t + h / 2)
42:         l2 = h * f2(x + k1 / 2, y + l1 / 2, t + h / 2)
43:         k3 = h * f1(x + k2 / 2, y + l2 / 2, t + h / 2)
44:         l3 = h * f2(x + k2 / 2, y + l2 / 2, t + h / 2)
45:         k4 = h * f1(x + k3, y + l3, t + h)
46:         l4 = h * f2(x + k3, y + l3, t + h)
47:         x += (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
48:         y += (1 / 6) * (l1 + 2 * l2 + 2 * l3 + l4)
49:     return T, X, Y
50:
51:
52: def main():
53:     h = 0.1
54:     g = 9.8
55:     l = 9.8
56:     q = 0.5
57:     fD = 1.2
58:     t_max = 100
59:     OmegaD = 2 / 3
60:     ep = 1e-6
61:     f1 = lambda theta, omega, t: omega
62:     f2 = lambda theta, omega, t: -(g / l) * np.sin(theta) - q * omega + fD * np.sin(OmegaD * t)
```

```
63:
64:     # Part a
65:     print("A")
66:     T1, Theta1, Omega1 = RungeKutta2(f1, f2, 0.2, 0, 0, t_max, h)
67:     T2, Theta2, Omega2 = RungeKutta2(f1, f2, 0.2 + ep, 0, 0, t_max, h)
68:     theta = [np.log(np.fabs(Theta1[i] - Theta2[i])) for i in range(len(Theta1))]
69:     pylab.plot(np.arange(0, t_max, h), theta)
70:     pylab.show()
71:
72:     # Part b
73:     print("B")
74:     theta = []
75:     for i in range(0, 100):
76:         print(">>{}".format(i))
77:         theta_0 = random.uniform(0.2, 0.21)
78:         T1, Theta1, Omega1 = RungeKutta2(f1, f2, theta_0, 0, 0, t_max, h)
79:         T2, Theta2, Omega2 = RungeKutta2(f1, f2, theta_0 + ep, 0, 0, t_max, h)
80:         if theta:
81:             theta = [
82:                 theta[i] + np.log(np.fabs(Theta1[i] - Theta2[i]))
83:                 for i in range(len(Theta1))
84:             ]
85:         else:
86:             theta = [
87:                 np.log(np.fabs(Theta1[i] - Theta2[i]))
88:                 for i in range(len(Theta1))
89:             ]
90:
91:     theta = [x / 100 for x in theta]
92:     E_x = 0.0
93:     E_y = 0.0
94:     E_xx = 0.0
95:     E_xy = 0.0
96:     for pt in zip(np.arange(0, t_max, h), theta):
97:         E_x += pt[0]
98:         E_y += pt[1]
99:         E_xx += (pt[0]**2)
100:        E_xy += (pt[0] * pt[1])
101:    N = len(np.arange(0, t_max, h))
102:    E_x /= N
103:    E_y /= N
104:    E_xx /= N
105:    E_xy /= N
106:    m = (E_xy - E_x * E_y) / (E_xx - E_x**2)
107:    c = (E_xx * E_y - E_x * E_xy) / (E_xx - E_x**2)
108:    print("Y={} * x + {}".format(m, c))
109:    pylab.plot(np.arange(0, t_max, h), theta)
110:    pylab.plot(
111:        np.arange(0, t_max, h), [m * x + c for x in np.arange(0, t_max, h)])
112:    pylab.show()
113:
114:
115: if __name__ == "__main__":
116:     main()
```