ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №16**

Выполнил(а) студент группы М8О-203Б-22

Вертоградский Д.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Авдюшкин А.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Москва, 2023

*Задание:* проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

Вариант 16.

*Реализация:*

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

def SystDiffEq(y, t, m=0.3, l=1, c=2, k1=1, k2=1, g=9.81):

dy = np.zeros\_like(y)

dy[0] = y[2]

dy[1] = y[3]

a11 = 1

a12 = -np.cos(y[0]-y[1])

b1 = y[3]\*\*2 \* np.sin(y[0]-y[1])+(g/l)\*np.sin(y[0])-(c\*y[0]+k1\*y[2])/(m\*l\*\*2)

a21 = -np.cos(y[0]-y[1])

a22 = 1

b2 = -y[2]\*np.sin(y[0]-y[1])-(g/l)\*np.sin(y[1])-k2\*y[3]/(m\*l\*\*2)

detA = a11 \* a22 - a12 \* a21

detA1 = b1 \* a22 - a12 \* b2

detA2 = a11 \* b2 - a21 \* b1

dy[2] = detA1 / detA

dy[3] = detA2 / detA

return dy

steps = 1000

t = np.linspace(0, 10, steps)

y0 = [np.pi/3, -np.pi/3, 0 ,0]

Y = odeint(SystDiffEq, y0, t)

phi = Y[:,0]

theta = Y[:,1]

phit = Y[:,2]

thetat = Y[:,3]

m=0.3

l=1

g=9.81

phitt = np.zeros\_like(t)

thetatt = np.zeros\_like(t)

for i in range(len(t)):

phitt[i] = SystDiffEq(Y[i], t[i])[2]

thetatt[i] = SystDiffEq(Y[i], t[i])[3]

Rx = m\*l\*(thetatt\*np.cos(theta)-thetat\*\*2\*np.sin(theta)-phitt\*np.cos(phi)+phit\*\*2\*np.sin(phi))

Ry = m\*g+m\*l\*(thetatt\*np.sin(theta)+thetat\*\*2\*np.cos(theta)-phitt\*np.sin(phi)-phit\*\*2\*np.cos(phi))

figrt = plt.figure()

phiplt = figrt.add\_subplot(4,1,1)

phiplt.plot(t, phi)

phiplt.set\_title("phi")

thetaplt = figrt.add\_subplot(4,1,2)

thetaplt.plot(t, theta)

thetaplt.set\_title("theta")

Rxplt = figrt.add\_subplot(4,1,3)

Rxplt.plot(t, Rx)

Rxplt.set\_title("Rx")

Ryplt = figrt.add\_subplot(4,1,4)

Ryplt.plot(t, Ry)

Ryplt.set\_title("Ry")

figrt.show()

figr = plt.figure()

gr = figr.add\_subplot(1,1,1)

gr.set\_xlim([-3.5, 2.5])

gr.set\_ylim([-2.5, 3.5])

gr.plot([-0.5, 0.5, 0, -0.5], [0.5, 0.5, 1, 0.5], linewidth=1)

lenOA = 2.25

lenAB = 2

xO = 0

yO = 1

xA = xO + lenOA \* np.sin(phi)

yA = yO + lenOA \* np.cos(phi)

xB = xA - lenAB \* np.sin(theta)

yB = yA - lenAB \* np.cos(theta)

pO = gr.plot(xO, yO)[0]

pA = gr.plot(xA[0], yA[0], marker='.')[0]

pB = gr.plot(xB[0], yB[0], marker='.')[0]

lineAB = gr.plot([xA[0], xB[0]], [yA[0], yB[0]], color='black')[0]

lineOA = gr.plot([xO, xA[0]], [yO, yA[0]], color='black')[0]

Ns = 3

r1 = 0.1

r2 = 0.4

numpnts = np.linspace(0, 1, 50\*Ns+1)

Betas = numpnts \* (Ns \* 2 \* np.pi - phi[0]+1.5)

Xs = (r1 + (r2 - r1) \* numpnts \* np.cos(Betas)) - 0.1

Ys = (r1 + (r2 - r1) \* numpnts \* np.sin(Betas)) - 0.1

SpPruzh = gr.plot(Xs + xO, Ys + yO)[0]

def run(i):

pA.set\_data(xA[i], yA[i])

pB.set\_data(xB[i], yB[i])

lineAB.set\_data([xA[i], xB[i]], [yA[i], yB[i]])

lineOA.set\_data([xO, xA[i]], [yO, yA[i]])

Betas = numpnts \* (Ns \* 2 \* np.pi - phi[i]+1.5)

Xs = (r1 + (r2 - r1) \* numpnts \* np.cos(Betas)) - 0.1

Ys = (r1 + (r2 - r1) \* numpnts \* np.sin(Betas)) - 0.1

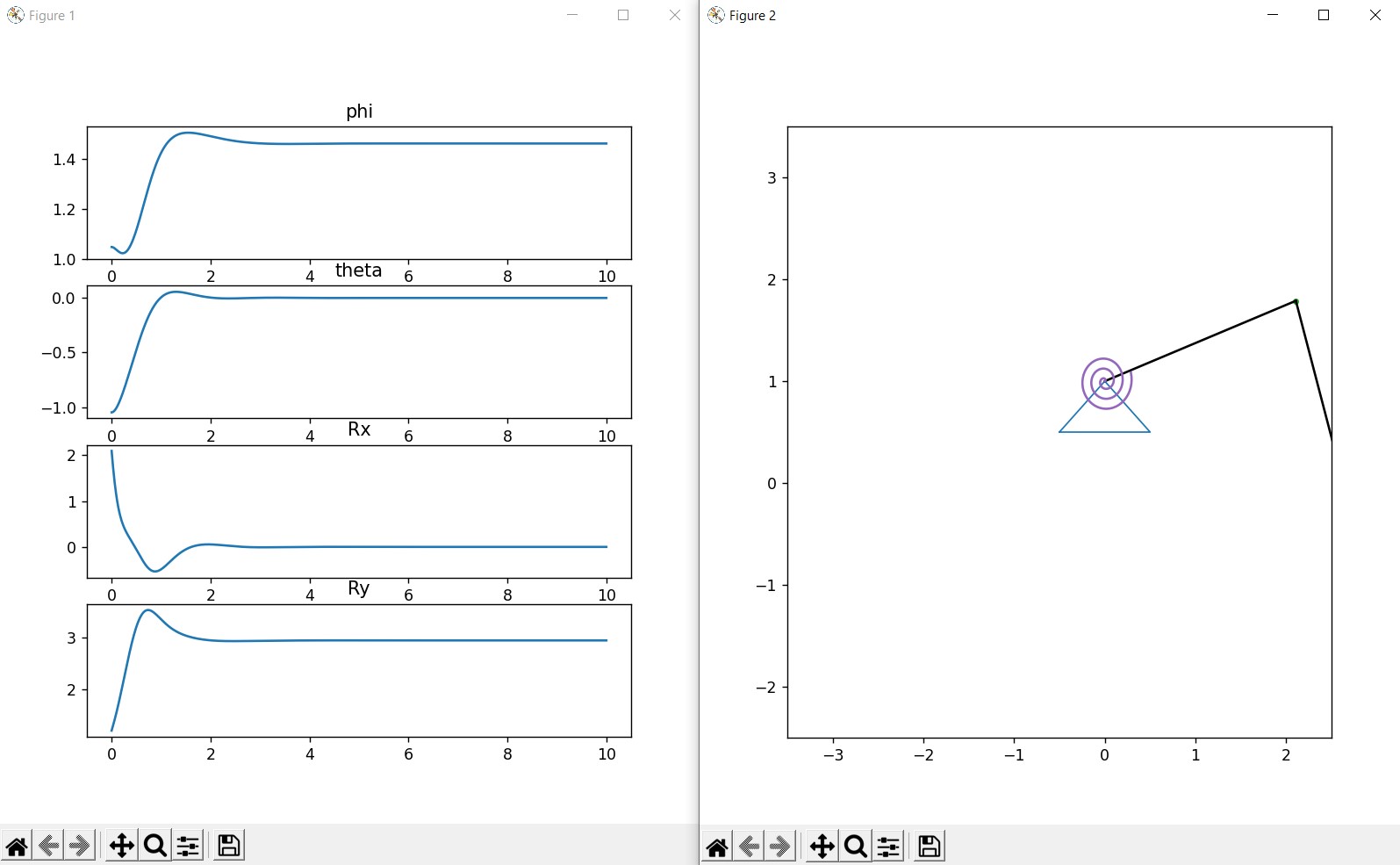
SpPruzh.set\_data(Xs + xO, Ys + yO)

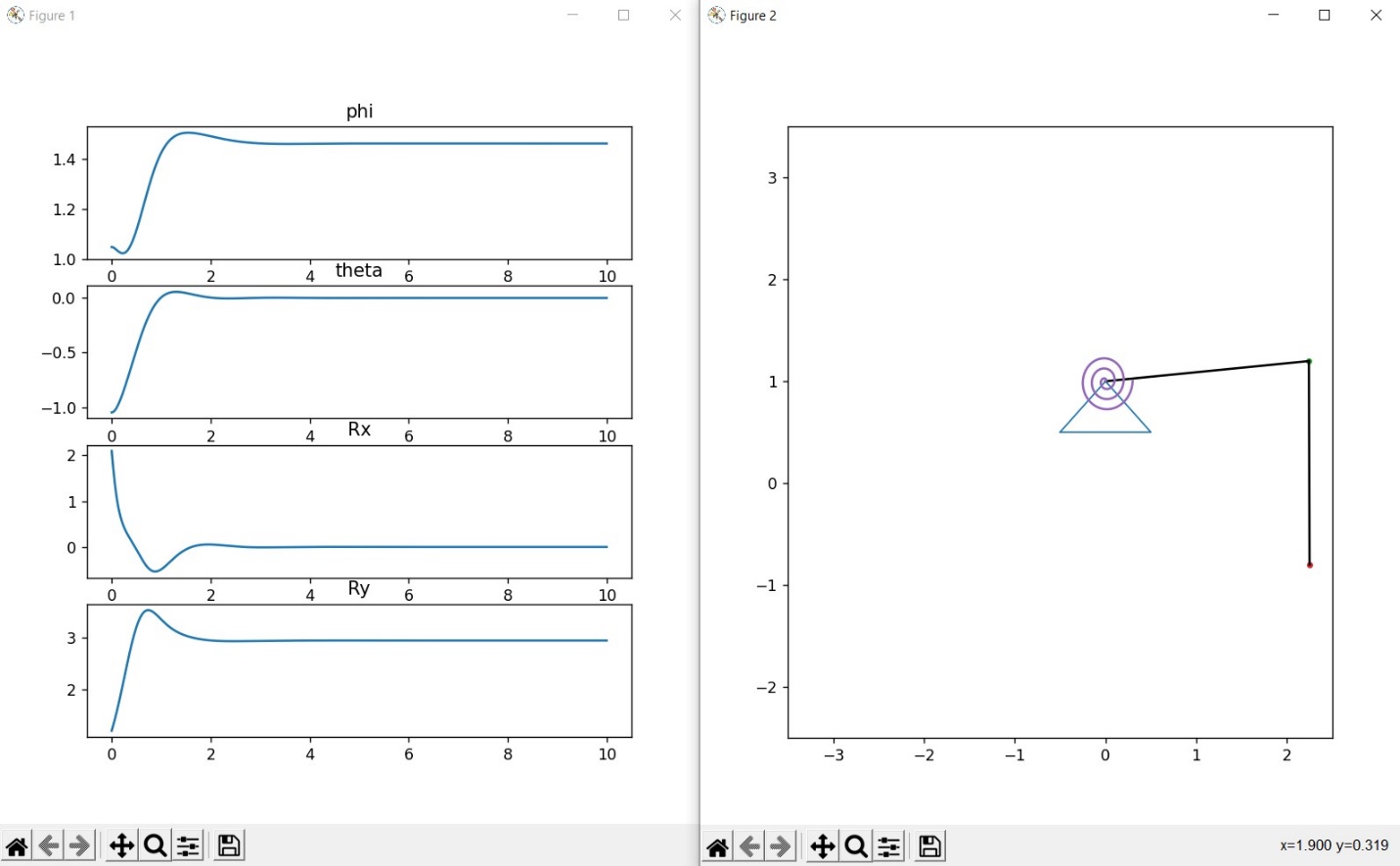
return

anim = FuncAnimation(figr, run, frames=steps, interval=0.5)

plt.show()

*Результаты:*





*Выводы:* Проинтегрирование системы дифференциальных уравнений движения с помощью Python позволяет получить численные значения координат и скоростей элементов системы для каждого шага времени.

Создание анимации движения системы позволяет наглядно представить и изучить динамику системы.

Построение графиков законов движения и реакций позволяет более детально изучить изменение координат и скоростей элементов системы во времени, а также оценить их взаимосвязь и зависимость от внешних факторов.

Проинтегрирование системы дифференциальных уравнений и создание анимации с помощью Python может быть полезно для анализа и предсказания движения объектов в реальных ситуациях, таких как движение тел в физических системах или траектории движения транспортных средств.

Таким образом, выполнение лабораторной работы позволяет более глубоко изучить и понять движение системы с двумя степенями свободы, а также оценить его характеристики и динамику.