ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

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

**ВАРИАНТ ЗАДАНИЯ № 14**

Выполнил(а) студент группы М8О-208Б-21

Марков И. И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

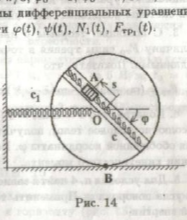
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Лабораторная работа 2.**

*Задание:* реализовать анимацию движения механической системы.

**1. Механическая система**

****

**2. Код программы**

import numpy as np

from math import pi

from matplotlib.animation import FuncAnimation

from matplotlib import pyplot as plt

# константы

t\_fin = 20

t = np.linspace(0, t\_fin, 1001)

x0 = 5 # положение кольца в начальный момент времени

R = 3 # радиус кольца

s = [0.9 \* R \* np.cos(2 \* x) for x in t] # отклонение груза

phi = [1 + 0.5\*np.cos(4 \* x + pi) for x in t] # угол поворота кольцы

angles = np.linspace(0, 2 \* pi , 360)

box\_w = 0.4 # ширина груза

box\_h = 0.2  # высота груза

# генерирует пружинку высотой h с количеством витков k и шириной w

def spring(k, h, w):

    x = np.linspace(0, h, 100)

    return np.array([

        x,

        np.sin(2 \* pi / (h / k) \* x) \* w

    ])

# массивы, где будем хранить просчитанные точки

ring\_dots\_x\_tmp = R \* np.cos(angles)

ring\_dots\_y\_tmp = R \* np.sin(angles)

box\_x\_tmp = np.array([-box\_h / 2, -box\_h / 2, box\_h / 2, box\_h / 2, -box\_h / 2])

box\_y\_tmp = np.array([-box\_w / 2, box\_w / 2, box\_w / 2, -box\_w / 2, -box\_w / 2])

ring\_dots\_x = np.zeros([len(t), len(angles)])

ring\_dots\_y = np.zeros([len(t), len(angles)])

box\_dots\_x = np.zeros([len(t), 5])

box\_dots\_y = np.zeros([len(t), 5])

spring\_a\_x = np.zeros([len(t), 100])

spring\_a\_y = np.zeros([len(t), 100])

spring\_b\_x = np.zeros([len(t), 100])

spring\_b\_y = np.zeros([len(t), 100])

spring\_c\_x = np.zeros([len(t), 100])

spring\_c\_y = np.zeros([len(t), 100])

# считаем точки

for i in range(len(t)):

    # центр кольца

    ring\_x = x0 + phi[i] \* R

    ring\_y = R

    # точки окружности кольца

    ring\_dots\_x[i] = np.cos(phi[i]) \* ring\_dots\_x\_tmp + np.sin(phi[i]) \* ring\_dots\_y\_tmp + ring\_x

    ring\_dots\_y[i] = - np.sin(phi[i]) \* ring\_dots\_x\_tmp + np.cos(phi[i]) \* ring\_dots\_y\_tmp + ring\_y

    # точки груза

    bx = box\_x\_tmp - s[i]

    by = box\_y\_tmp

    box\_dots\_x[i] = np.cos(phi[i]) \* bx + np.sin(phi[i]) \* by + ring\_x

    box\_dots\_y[i] = - np.sin(phi[i]) \* bx + np.cos(phi[i]) \* by + ring\_y

    # горизонтальная пружина

    spring\_a\_x[i] = spring(5, ring\_x, 0.2)[0]

    spring\_a\_y[i] = spring(5, ring\_x, 0.2)[1] + ring\_y

    # верхняя пружина в кольце

    b\_x = R - spring(10, R + s[i] - box\_h / 2, 0.2)[0]

    b\_y = spring(10, R - s[i], 0.2)[1]

    spring\_b\_x[i] = np.cos(phi[i]) \* b\_x + np.sin(phi[i]) \* b\_y + ring\_x

    spring\_b\_y[i] = -np.sin(phi[i]) \* b\_x + np.cos(phi[i]) \* b\_y + ring\_y

    # нижняя пружина в кольце

    c\_x = spring(10, R - s[i] - box\_h / 2, 0.2)[0] - R

    c\_y = spring(10, R - s[i], 0.2)[1]

    spring\_c\_x[i] = np.cos(phi[i]) \* c\_x + np.sin(phi[i]) \* c\_y + ring\_x

    spring\_c\_y[i] = -np.sin(phi[i]) \* c\_x + np.cos(phi[i]) \* c\_y + ring\_y

# рисуем график

fig = plt.figure() # создаем холст, на котором будем рисовать фигуры

ax = fig.add\_subplot(1, 1, 1)

ax.axis("equal")

surface = ax.plot([0, 0, 10], [10, 0, 0]) # пол и стена

ring, = ax.plot(ring\_dots\_x[0], ring\_dots\_y[0]) # кольцо

box, = ax.plot(box\_dots\_x[0], box\_dots\_y[0]) # груз

spring\_a, = ax.plot(spring\_a\_x[0], spring\_a\_y[0]) # горизонтальная пружина

spring\_b, = ax.plot(spring\_b\_x[0], spring\_b\_y[0]) # нижняя пружина

spring\_c, = ax.plot(spring\_c\_x[0], spring\_c\_y[0]) # верхняя пружина

# функция анимации

def animate(i):

    ring.set\_data(ring\_dots\_x[i], ring\_dots\_y[i])

    box.set\_data(box\_dots\_x[i], box\_dots\_y[i])

    spring\_a.set\_data(spring\_a\_x[i], spring\_a\_y[i])

    spring\_b.set\_data(spring\_b\_x[i], spring\_b\_y[i])

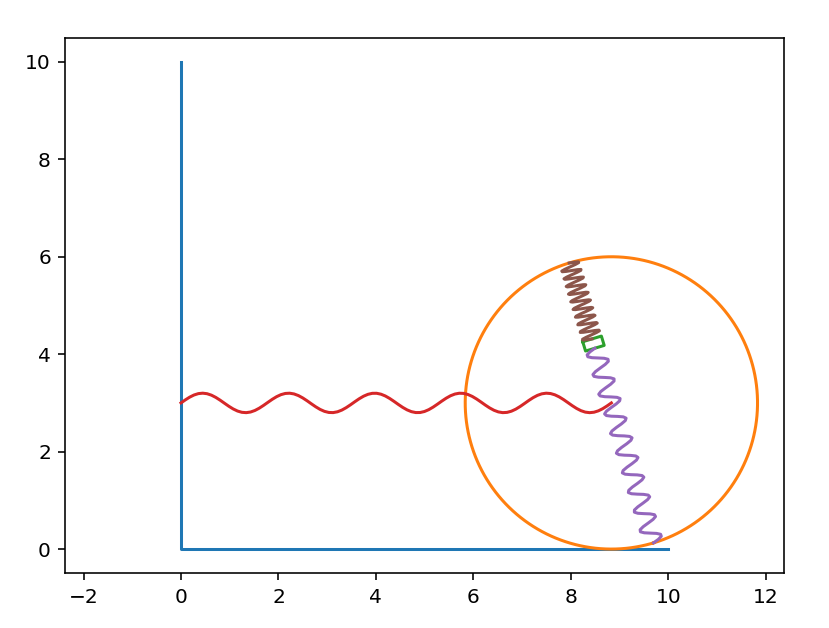
    spring\_c.set\_data(spring\_c\_x[i], spring\_c\_y[i])

    return ring, box, spring\_a, spring\_b, spring\_c

animation = FuncAnimation(fig, animate, frames=len(t), interval=60)

plt.show()

**3. Результат работы программы**

****