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 №30

Выполнил(а) студент группы М8О-208Б-20

Ханнанов Руслан Маратович_____
подпись, дата

Проверил и принял

Доцент каф. 802, Чекина Е.А._____
подпись, дата

с оценкой _____

Москва, 2021

Вариант №«30»

Задание:

Реализовать анимацию движения механической системы используя язык программирования Python.

Механическая система:

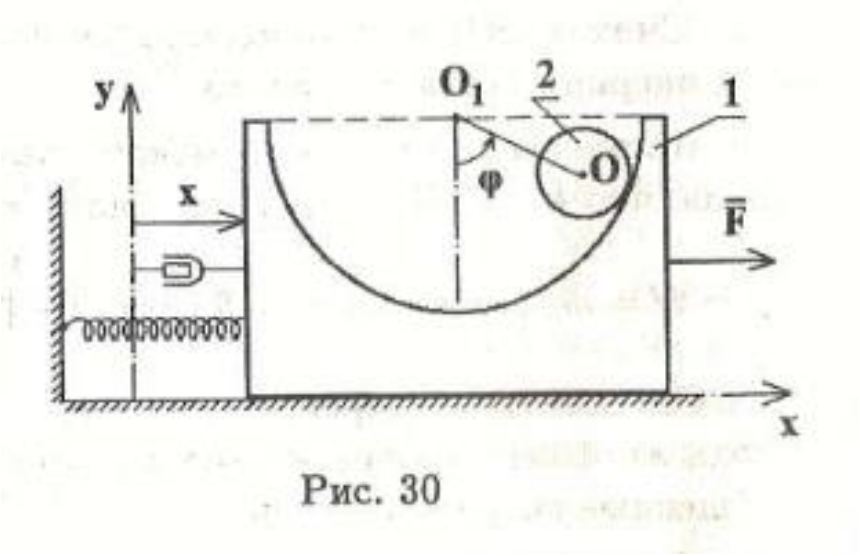


Рис. 30

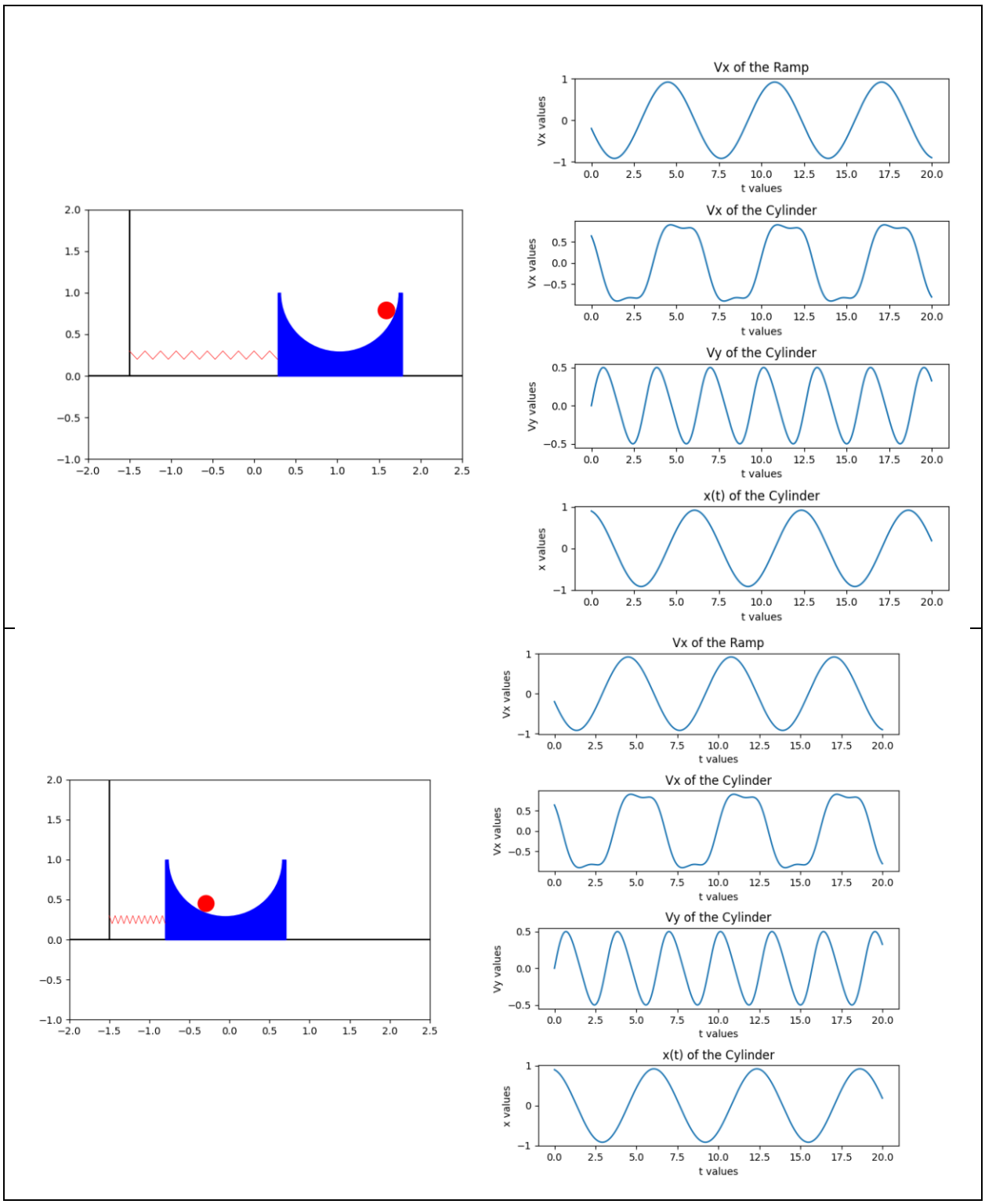
Текст программы:

```
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import numpy as np
import math
from matplotlib.animation import FuncAnimation
import sympy as sp

if __name__ == '__main__':

    t = sp.Symbol('t')

    # Начальный параметры
    x_0 = 0.1
    d = 0.7 - 0.1
    O1_x = 0.75
    O1_y = 1.0

    # Вручную заданные фукнции угла и положения
    alpha = 80 * sp.sin(t) * math.pi / 180
    Rec_x = 0.9 * sp.cos(t) - 0.2 * sp.sin(t)

    o_y = O1_y - d * sp.cos(alpha)
    o_x = O1_x + d * sp.sin(alpha) + Rec_x
```

```
    VxO = sp.diff(o_x, t)
    VyO = sp.diff(o_y, t)

    VxRec = sp.diff(Rec_x, t)

    # Массивы
    T = np.linspace(0, 20, 1000)
    ALPHA = np.zeros_like(T)
    OY = np.zeros_like(T)
    OX = np.zeros_like(T)
    REC_X = np.zeros_like(T)
    VXREC = np.zeros_like(T)
    VXO = np.zeros_like(T)
    VYO = np.zeros_like(T)

    for i in np.arange(len(T)):
        ALPHA[i] = sp.Subs(alpha, t, T[i])
        REC_X[i] = sp.Subs(Rec_x, t, T[i])
        OY[i] = O1_y - d * sp.cos(ALPHA[i])
        OX[i] = O1_x + d * sp.sin(ALPHA[i]) + REC_X[i]
        VXREC[i] = sp.Subs(VxRec, t, T[i])
        VXO[i] = sp.Subs(VxO, t, T[i])
        VYO[i] = sp.Subs(VyO, t, T[i])

    # Границы рисунка
    #fig, ax = plt.subplots()
    fig = plt.figure(figsize=(4, 10))
    ax = fig.add_subplot(1, 2, 1)
    plt.xlim(-2, 2.5)
    plt.ylim(-1, 2)
    ax.set_aspect(1)

    # Прямоугольник
    body1 = plt.Rectangle((0.0, 0.0), width=1.5, height=1.0, color='b')

    # Линия нижней поверхности
    bottom_line_x = [-2, 2.5]
    bottom_line_y = [0, 0]
    plt.plot(bottom_line_x, bottom_line_y, 'k')

    # Линия боковой поверхности
    side_line_x = [-1.5, -1.5]
    side_line_y = [0, 2]
    plt.plot(side_line_x, side_line_y, 'k')

    # Выколотая окружность
    white_circle = plt.Circle((0.75, 1.0), radius=0.7, color='w')

    # Функция создает набор координат x,y для построения зигзагообразной
линии, которая изображает пружину
    def get_spring_line(length, coils, diameter):
        x = np.linspace(0, length, coils * 2)
        y = [diameter * 0.5 * (-1) ** i for i in range(len(x))]
        return np.array([x, y])

    # Пружина
    spring_xy = get_spring_line(1.5, 10, 0.1)
    spring = mlines.Line2D(spring_xy[0] - 1.5, spring_xy[1] + 0.25, lw=0.5,
color='r')

    # Цилиндр
    cylinder = plt.Circle((OX[0], OY[0]), radius=0.1, color='r')

    # Графики
```

```
    ax2 = fig.add_subplot(4, 2, 2)
    ax2.plot(T, VXREC)
    plt.title('Vx of the Ramp')
    plt.xlabel('t values')
    plt.ylabel('Vx values')

    ax3 = fig.add_subplot(4, 2, 4)
    ax3.plot(T, VXO)
    plt.title('Vx of the Cylinder')
    plt.xlabel('t values')
    plt.ylabel('Vx values')

    ax4 = fig.add_subplot(4, 2, 6)
    ax4.plot(T, VYO)
    plt.title('Vy of the Cylinder')
    plt.xlabel('t values')
    plt.ylabel('Vy values')

    ax5 = fig.add_subplot(4, 2, 8)
    ax5.plot(T, REC_X)
    plt.title('x(t) of the Cylinder')
    plt.xlabel('t values')
    plt.ylabel('x values')

    # ax6 = fig.add_subplot(2, 1, 2)
    # ax6.plot(T, ALPHA)
    # plt.title('alpha(t) of the Cylinder')
    # plt.xlabel('t values')
    # plt.ylabel('alpha values')

    plt.subplots_adjust(wspace=0.3, hspace=0.7)

    def init():
        # Прямоугольник
        ax.add_patch(body1)
        # Выколотая окружность
        ax.add_patch(white_circle)
        # Пружина
        ax.add_line(spring)
        # Цилиндр
        ax.add_patch(cylinder)
        return body1, white_circle, spring, cylinder

    def anima(j):  # Анимация движения
        cylinder.center = OX[j], OY[j]
        white_circle.center = REC_X[j] + 0.75, 1.0
        body1.xy = REC_X[j], 0
        sp_xy = get_spring_line(1.5 + REC_X[j], 10, 0.1)
        spring.set_data(sp_xy[0] - 1.5, sp_xy[1] + 0.25)
        return body1, white_circle, spring, cylinder


    # Анимация
    anim = FuncAnimation(fig, anima, init_func=init, frames=1000,
interval=2.5, blit=True)
    plt.show()
```

Результат работы:



Вывод:

В этой лабораторной работе я научился началам моделирования сложной механической системы, пока не используя физических законов. Научился работать с Python библиотеками, такими как: matplotlib, numpy и sympy.